Universidad Carlos III de Madrid

Control de Software Educativo Mediante Kinect de Microsoft

Trabajo Fin de Grado

AUTOR: Estefanía Fernández Sánchez

TUTOR: Telmo Zarraonandia Ayo

5-9-2012

Título: Control de Software Educativo Mediante Kinect de Microsoft

Autor: Estefanía Fernández Sánchez

Tutor: Telmo Zarraonandia Ayo

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ___ de _____ de 2012 en Leganés, en la escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Quiero dedicar en primer lugar y de forma muy especial este proyecto a mis padres, Luisa y Pedro, porque si no fuera por ellos no estaría hoy aquí escribiendo estas líneas, porque su esfuerzo y dedicación han hecho que fuera mejor persona y que haya intentado mejorar día a día, no solo a nivel personal, sino también profesional. Esto es para vosotros, para que veáis que todo esfuerzo tiene su recompensa.

A mi hermana Laura, porque aunque no nos llevemos muy bien y estemos siempre discutiendo, sabe que la quiero, que esto son solo cosas que pasan entre hermanos, que es algo normal.

Al resto de mi familia, que aunque no os pueda nombrar uno a uno y ahora vivamos un poco más lejos, siempre que me ven me preguntan qué tal me va con la carrera y me dan ánimos para continuar.

Quiero agradecer también a toda esa gente maravillosa que he conocido a lo largo de estos cuatro años, porque de esta experiencia no me llevo solo conocimientos y un título, sino que también me llevo muchas personas increíbles, muchos amigos que sé que estarán ahí siempre. Marta, Lisardo, Raquel, Josito, Miguel, Nerea, Álvaro, Gabriel, Javi, Rodry… Sois tantos… Y, por supuesto, no me he olvidado de ti, Bea, pero creo que te mereces una mención especial, por todos los momentos que he pasado contigo, por los días de prácticas hasta las tantas, porque tú me entiendes como nadie y siempre sabes qué me pasa, porque si me llevo algo importante de esta época en la universidad, sin duda es una amiga como tú. Simplemente, gracias por todo.

A Irene, porque da igual las cosas que hayan pasado, porque aunque hayamos tenido nuestros más y nuestros menos, estos más de dos años que hace que te conozco han sido increíbles. Gracias por tu comprensión, por tu paciencia, por tus ánimos cuando casi me sentía derrotada, gracias por estar ahí siempre, por tu cariño y por ser capaz de sacarme siempre una sonrisa con un simple ‘hola’.

A mis compañeros de la biblioteca, a los que están y a los que ya se fueron, a los becarios y a los funcionarios, porque vosotros también habéis formado parte de esta etapa universitaria y me habéis regalado grandes momentos. A todos, Regina, Ángela, Lucia, Fer, Ángel, David, Javi, Rocío, Héctor, Manu, Laura, Leonor y a los que se me olviden, muchas gracias.

Gracias a mi tutor, Telmo, por la confianza depositada en mí para la realización de este proyecto, por su paciencia y comprensión con tanto retraso y cambio de fechas, por todo el esfuerzo y dedicación que ha puesto para que el proyecto saliese adelante de forma satisfactoria.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN Y OBJETIVOS

1.1. INTRODUCCIÓN

Dado el continuo avance de la tecnología, han ido surgiendo nuevos periféricos de entrada de datos que nos permiten interactuar con los ordenadores. La interacción más clásica se lleva a cabo con el teclado y el ratón del ordenador, aunque existen otros periféricos como micrófonos, escáneres, cámaras de fotos o webcams que nos permiten proporcionar una entrada de datos al ordenador. El desarrollo de la tecnología, nos permite actualmente tener otro tipo de interacción, como puede ser el caso de las pantallas táctiles, cuyo uso está muy extendido no solo en el ámbito de los ordenadores personales, sino también en los dispositivos móviles, como ordenadores portátiles, PDA's o Smartphones. Si vamos un paso más adelante, podemos encontrar periféricos como el WiiMote o el PlayStation Move que, a pesar de estar creados inicialmente para la interacción con videoconsolas, cada vez más son utilizados para comunicarnos con un ordenador mediante el uso de los propios gestos del usuario, es decir, con su propio movimiento.

Es en este punto donde encontramos la Interfaz Natural de Usuario (o NUI, *Natural User Interface*), un sistema que nos permite interactuar con las aplicaciones sin necesidad de utilizar mandos o dispositivos de entrada de datos como el ratón o el teclado, sino que se basa en movimientos gestuales de las manos o el propio cuerpo como forma de comunicación. Un claro ejemplo es el uso de las pantallas táctiles que utilizan las yemas de los dedos como forma de interacción, o el WiiMote y el PlayStation Move antes mencionados que, a pesar de que en estos casos es necesario el uso de un mando, nos permiten comunicarnos con el ordenador mediante los movimientos de las manos y los brazos. Además de los gestos, se está utilizando también el uso de la voz como forma de interacción dentro de la NUI.

En 2010 ve la luz el sensor Kinect, un periférico capaz de reconocer un esqueleto completo y, por tanto, los movimientos realizados con el mismo. Kinect nos proporciona un reconocimiento en 3D, de modo que es capaz de medir las distancias y contemplar los movimientos en los tres ejes. Este dispositivo, ligado a la cantidad de posibilidades que ofrece, supone una auténtica revolución en el entorno de la NUI y en la forma de comunicarnos con el ordenador. Entre las posibilidades que nos ofrece Kinect podemos destacar:

- **TedCas.** Se trata de una aplicación desarrollada para el sector de la salud, que permite a médicos y enfermeras acceder a información digital mediante el uso de gestos y comandos de voz, evitando de este modo contaminaciones a través de ratones, teclados o pantallas táctiles en el entorno de los hospitales, sobre todo en quirófanos. (TedCas)
- **VirtualRehab.** VirtualRehab es una aplicación para la rehabilitación de pacientes con algún grado de discapacidad física (Innodevices).
- **Kinecthesia.** El proyecto Kinecthesia trata de ayudar a personas con incapacidad visual detectando los objetos y devolviendo la información por medio de vibraciones al usuario invidente (WebAdictos, 2011).

Este fenómeno es denominado “Efecto Kinect” y pueden encontrarse aplicaciones desde ayuda a niños con autismo a apoyo a cirujanos en quirófano, pasando por control de robots o el propio ordenador sin necesidad de un ratón.

Entonces, ¿por qué usar Kinect y no otro dispositivo? Kinect, como hemos podido observar nos proporciona un mayor abanico de posibilidades. Además, existe una gran comunidad de desarrolladores para el dispositivo, lo que hace que se encuentre en continuo avance y nos ofrezca cada vez más opciones. Sobre todo, Kinect nos permite olvidarnos de cualquier tipo de mando para realizar la interacción con el computador. Concretamente, en el ámbito de la educación que es en el que se trabajará, permitirá al profesor moverse con total libertad por la clase, cambiar el modo de visión de la aplicación con un simple gesto e incluso pasar las diapositivas de un PowerPoint sin la necesidad de volver al ordenador para hacerlo, ahorrando así tiempo aprovechable en continuar con la explicación.

1.2. OBJETIVOS

El presente trabajo se engloba de un proyecto de mayor magnitud consistente en una aplicación de realidad aumentada para el ámbito educativo (Zarraonandia, Francese, Passero, Díaz, & Tortora, 2011). Esta aplicación ofrece un *feedback* continuo del estado de los alumnos durante una clase, de forma que el profesor pueda conocer si los alumnos han entendido la explicación, si han conseguido resolver un determinado problema propuesto o si tienen algún tipo de duda durante la clase. Este proyecto global dispone de diferentes módulos, entre los que se encuentran la propia aplicación de realidad aumentada, un sistema basado en el uso de smartphones que permite que los alumnos proporcionen este *feedback* al profesor, y un controlador que permita al profesor manejar la aplicación educativa mediante NUI (*Interfaz Natural de Usuario*), es decir, mediante gestos con el propio cuerpo. De esta última parte es sobre la que trataremos a lo largo de este documento.

Por lo tanto, el principal objetivo de este proyecto es crear una aplicación capaz de reconocer una serie de gestos básicos con los que controlar una herramienta educativa. Además de estos gestos, la aplicación deberá permitir a los usuarios decidir que gestos van a utilizar para cada una de las acciones que se pueden realizar. Además, al finalizar este proyecto seremos capaces de controlar las diapositivas de un PowerPoint utilizando los gestos reconocidos.

Al margen de este objetivo principal, tenemos otra serie de objetivos derivados de este. Así, deberemos aprender qué tecnologías existen en la actualidad y cuáles fueron sus antecedentes, de manera que podamos decidir cuál de ellas es la más apropiada para llevar a cabo las acciones anteriormente mencionadas.

Una vez decidida la tecnología a utilizar deberemos aprender qué es y cómo funciona. De esta forma conoceremos en profundidad de que partes dispone y cuál es el funcionamiento del sensor Kinect. Con este conocimiento nos será más sencillo entender el desarrollo de aplicaciones para este dispositivo.

Otro punto importante es conocer las librerías de desarrollo existentes para la creación de aplicaciones para el sensor Kinect, concretamente la SDK oficial de Microsoft que será la utilizada para el desarrollo del proyecto. Con ello tendremos conocimiento de cuáles son los requisitos técnicos y qué debemos saber para comenzar a desarrollar aplicaciones para Kinect, así como la cantidad de posibilidades que nos ofrece.

1.3. FASES DEL DESARROLLO

En este punto se comentarán las diferentes fases de desarrollo llevadas a cabo para la correcta elaboración del mismo.

- **Estudio de viabilidad.** En este punto se analizarán las diferentes opciones de las que disponemos para llevar a cabo el proyecto. Se compararán los diferentes dispositivos que pueden ser utilizados, las librerías de desarrollo existentes o los lenguajes de programación que pueden ser utilizados, con el fin de fijar los elementos a utilizar.
- **Documentación previa.** Una vez conocemos el hardware a utilizar y el entorno de desarrollo, lenguajes y librerías que serán usados para la realización del proyecto, se procederá a la documentación. Esta documentación previa es importante ya que se trata de una tecnología bastante novedosa de la que no se tienen conocimientos previos, de modo que consigamos una base teórica sobre el desarrollo del proyecto.
- **Análisis.** En esta fase se procederá a analizar los casos de uso y requisitos de usuario teniendo en cuenta la funcionalidad que debe tener la aplicación una vez finalizado el desarrollo.
- **Diseño.** En este punto se tratará de resolver el problema planteado en el análisis especificando los componentes del sistema, así como las relaciones entre ellos y la arquitectura utilizada.
- **Implementación.** Una vez tenemos el análisis y el diseño realizado, pasaremos a realizar la implementación. Al finalizar esta fase, dispondremos de una aplicación con toda la funcionalidad descrita en las fases anteriores.
- **Pruebas.** En esta fase se realizarán las pruebas que se considere oportunas para comprobar que el software funciona de forma correcta y que han sido contempladas todas las funcionalidades descritas en el diseño y el análisis de requisitos.
- **Documentación.** En este punto se realizará la documentación necesaria para la total comprensión del proyecto. De este modo, se incluirán diagramas, manuales de usuario u otros datos que se considere oportuno.

1.4. MEDIOS EMPLEADOS

A continuación describiremos brevemente los medios empleados, tanto hardware como software, para la realización del presente proyecto.

1.4.1. Elementos Hardware

Los elementos hardware utilizados son los detallados a continuación:

- **Ordenador.** Se ha utilizado un ordenador portátil tanto para el desarrollo de la aplicación, como para la elaboración de la documentación relacionada. El modelo utilizado ha sido un Packard Bell EasyNote TJ76 con procesador Intel Core i5, 4 GB de memoria RAM y sistema operativo Windows 7 Home Premium.
- **Sensor Kinect.** Se ha utilizado el sensor Kinect para XBOX 360, debido a que al inicio del proyecto no había salido a la venta el Kinect para Windows. En cuanto a los detalles del sensor, se verán en profundidad a lo largo del documento.
- **Cable de alimentación Kinect.** Para que el sensor funcione correctamente en el ordenador, se debe disponer de un adaptador que conecte Kinect a un puerto USB del pc, así como a una alimentación externa. Esto es debido a que el puerto USB no proporciona la suficiente energía al sensor Kinect para que podamos hacer uso de todo su potencial.

1.4.2. Elementos Software

Los elementos software utilizados para el desarrollo del proyecto son los siguientes:

- **Kinect for Windows SDK v1.0.** Librería de desarrollo oficial de Microsoft para el desarrollo de aplicaciones para el sensor Kinect. Incluye drivers y documentación técnica para la implementación de aplicaciones, APIs de referencia y documentación para la programación y una serie de ejemplos que muestran las buenas prácticas para el uso del sensor Kinect.
- **Microsoft Visual C# 2010 Express.** Entorno de desarrollo completo para la programación de aplicaciones en C#, lenguaje de programación utilizado en el proyecto para realizar la aplicación.
- **Microsoft Office 2010.** Se ha utilizado Microsoft Office como herramienta para la elaboración del presente documento.
- **Dropbox.** Servicio de alojamiento de archivos multiplataforma en la nube. Se ha utilizado para almacenar los archivos relativos al proyecto.
- **GanttProject.** Herramienta open source para la creación y gestión de diagramas de Gantt utilizados para la planificación del proyecto.

1.5. ESTRUCTURA DE LA MEMORIA

El presente documento ha sido organizado en diferentes capítulos, de forma que en cada uno de ellos tratemos un tema específico, haciendo que la memoria tenga una mejor organización y sea más fácil de leer y comprender. A continuación se explicará brevemente el contenido de cada uno de estos capítulos.

En el ***primer capítulo***, se expondrá una introducción en la que se mostrará la motivación por la que se eligió este proyecto, los objetivos que se quieren conseguir una vez la aplicación esté implementada, las diferentes fases del proyecto y los medios que han sido necesarios para la elaboración del mismo. De esta forma, conoceremos de forma breve y concisa en que consiste el proyecto.

En el ***segundo capítulo***, realizaremos un análisis de los dispositivos actuales que podrían ser utilizados para la realización del proyecto a modo de estado del arte. También veremos algunos de los dispositivos que son considerados antecesores de los analizados y finalmente justificaremos la elección del dispositivo realizando una comparación entre ellos. Del mismo modo se procederá con las diferentes librerías existentes para el desarrollo de aplicaciones para Kinect, justificando nuestra elección mediante una comparativa.

En el ***tercer capítulo***, se describirá detalladamente qué es el sensor Kinect y cuál es su funcionamiento básico. Además veremos cuáles son sus características técnicas y sus recomendaciones de uso.

En el ***cuarto capítulo***, se verá de forma detalla el SDK de Kinect, de forma que conozcamos cuáles son sus requerimientos tanto hardware como software y que conocimientos debemos tener para poder comenzar a desarrollar aplicaciones para Kinect. Además conoceremos cuál es su arquitectura, cómo se instala y la NUI API y Audio API que contiene.

En el ***quinto capítulo***, se expondrá el análisis, diseño e implementación de la aplicación, pudiendo considerarse de este modo el núcleo central del proyecto. En este capítulo se analizarán los casos de uso, requisitos del proyecto y la arquitectura utilizada, así como la implementación de los mismos.

En el ***sexto capítulo***, se analizarán las pruebas realizadas y los resultados obtenidos con el fin de ver si la aplicación funciona de forma correcta y se han tenido en cuenta todas las funcionalidades descritas en el análisis y diseño de la aplicación.

El ***séptimo capítulo***, expondrá las conclusiones alcanzadas una vez finalizado el proyecto. Veremos si los objetivos indicados al inicio de este documento han sido cumplidos y las impresiones finales obtenidas tras el desarrollo.

En el ***octavo capítulo***, estudiaremos las posibles mejoras que podrían implementarse en el futuro, nuevas funcionalidades, mejoras o evolución del proyecto.

En el ***noveno capítulo***, se incluirá información acerca de la gestión del proyecto, como la planificación del mismo o el presupuesto.

Para finalizar, se incluirán dos apartados. Por un lado, dispondremos de un ***glosario de términos***, con el fin de que el lector pueda consultarlo en caso de no conocer alguno de los acrónimos utilizados durante el documento. Por otro lado, se incluye una ***bibliografía*** con enlaces de interés sobre el proyecto.

2. ESTADO DEL ARTE

En este punto analizaremos brevemente las diferentes posibilidades que podemos encontrar para el desarrollo de este proyecto. En primer lugar, veremos qué periféricos de entrada de datos existen. A continuación, analizaremos qué librerías existen para el desarrollo de aplicaciones para el sensor Kinect.

2.1. PERIFÉRICOS DE ENTRADA

A continuación, trataremos dos vertientes diferentes, por un lado realizaremos un análisis retrospectivo, es decir, analizaremos algunos periféricos antiguos que pueden ser considerados como los antecesores de los que tenemos actualmente. Por otro lado, se analizaran los periféricos más actuales y realizaremos una comparativa con el fin de justificar las decisiones tomadas.

2.1.1. Periféricos Antiguos

2.1.1.1. EyeToy

EyeToy es una cámara de reconocimiento de movimiento que se conecta habitualmente a una videoconsola PlayStation 2 mediante un cable USB. Mediante el uso de esta cámara, un usuario puede interactuar con los juegos moviendo cualquier parte del cuerpo, principalmente con movimientos de los brazos. El jugador no tiene contacto físico con la tecnología, sino que la interacción es conducida por los movimientos mediante coincidencias en el espacio en un momento de tiempo determinado con botones y eventos de los juegos que el jugador puede ver en una imagen proyectada de su cuerpo en el centro de la pantalla.

La precisión de los movimientos viene dada por la resolución de la cámara, así como por el procesamiento de las tasas de frames. La cámara está adaptada para reconocer objetos específicos en ambientes particulares y no es capaz de adaptarse a entornos diferentes o cambios de iluminación. Esta tecnología únicamente permite detectar movimientos en el plano x-y, por lo que no detecta la profundidad como un movimiento en el plano z.

ILUSTRACIÓN 1. EYETOY

A pesar de que su uso fue principalmente para la interacción entre los usuarios y los juegos de la plataforma PlayStation 2, el dispositivo fue también utilizado como webcam en entornos Windows mediante la instalación de un driver determinado. (Twenebowa Larssen, Loke, Robertson, & Edwards, 2004)

2.1.1.2. Sega Activator

El Sega Activator es un anillo en forma de octógono que se coloca en el suelo y se conecta al puerto de la videoconsola Génesis de Sega. El producto se basa en el Light Harp creado por el artista y músico Assaf Gurner, y producido por Light y Sega.

El anillo emite rayos infrarrojos directamente hacia arriba, siendo luego reflejados desde el techo. Los movimientos del jugador, que se encuentra en el centro del anillo, hacen que estos rayos se corten, produciendo hasta 16 entradas distintas, utilizando manos y pies.

ILUSTRACIÓN 2. SEGA ACTIVATOR

En la comercialización del Activator se presentaba el terminal como un dispositivo de captura de movimiento de campo libre, es decir, que podría reconocer prácticamente cualquier movimiento realizado. De hecho, en los anuncios se muestran jugadores realizando movimientos de artes marciales en el anillo mientras se ve su traducción en el juego. Sin embargo, el sistema fue diseñado en realidad, sólo para que el jugador pudiera realizar movimientos específicos que eran análogos a los botones que se pulsan en un controlador cualquiera. El jugador es libre de dar puñetazos y patadas, pero solo se registrarán correctamente si se hace sobre el haz de luz correcto.

El Activator puede trabajar con cualquier juego de Génesis emulando un pad de tres botones y el pad de direcciones y asignándolos a cada una de las secciones del anillo, por lo que su uso puede llegar a resultar bastante incómodo. Además, no soporta los movimientos combinados, es decir, el pulsar dos botones a la vez. Únicamente tres juegos fueron programados específicamente para reconocer la gama de entradas del Activator.

Un gran problema de este dispositivo es que los techos abovedados, los espejos, techos irregulares, ventiladores en el techo y otras obstrucciones similares interrumpen las reflexiones de los rayos infrarrojos. Además, el Activator requiere de su propia fuente de alimentación. Debido a su alto coste, la falta de soporte y la jugabilidad restrictiva, sus ventas no fueron satisfactorias. (http://www.giantbomb.com/)

2.1.1.3. VRU (Voice Recognition Unit)

La VRU fue lanzada en el año 2000 por Nintendo como un complemento para el hardware de la consola Nintendo 64. Llegó en dos partes: la verdadera "unidad de reconocimiento de voz", que era una caja gris que se conecta a un puerto de control de la parte frontal de la Nintendo 64, y que codifica la voz en señales que el software del juego pueda entender; y un micrófono que se conecta a una ranura de la VRU.

ILUSTRACIÓN 3. VRU (VOICE RECOGNITION UNIT)

La VRU solo tenía dos juegos compatibles: *Hey you, Pikachu!* Y *Densha de Go! 64*. La VRU fue calibrada para un mejor reconocimiento de una voz aguda, como la voz de un niño. Como resultado, es menos probable que las voces de personas adultas y/o adolescentes fueran reconocidas correctamente.

Otro problema de la VRU es que no son compatibles entre regiones, es decir, la VRU japonesa no puede utilizarse en juegos de EEUU y viceversa, ya que las VRUs no serían detectadas por los juegos. (Nintendo)

2.1.2. Periféricos Actuales

2.1.2.1. WiiMote

Lanzado en diciembre de 2006, el WiiMote es una interfaz de control inercial para video juegos diferente a lo que se conocía hasta el momento. La principal innovación es su capacidad para localizarse dentro de dos grados de libertad tranlacionales y tres grados de libertad rotacionales. Esta localización se realiza con un buen grado de precisión, además de ser atractivo y económico.

La localización rotacional se realiza con la ayuda de tres sensores inerciales (acelerómetros/gravímetros) que miden la dirección de la gravedad a lo largo de los ejes. Por su parte, la localización tranlacional se realiza mediante la triangulación por luz infrarroja emitida por un sensor externo. Además, el WiiMote dispone de 12 botones tradicionales que pueden ser utilizados como complemento para su localización.

ILUSTRACIÓN 4. WIIMOTE

El WiiMote se comunica con otros dispositivos utilizando Bluetooth. Existen ciertos eventos que hacen que el WiiMote mande un paquete de actualización de estado al dispositivo conectado. Algunos de estos eventos son: botón presionado, cambio en los datos de los acelerómetros o el sensor de infrarrojos, y los cambios en los dispositivos de extensión del WiiMote. El Nunchuk es una de estas extensiones, que se conecta físicamente con el WiiMote y agrega un segundo conjunto de tres acelerómetros junto con dos botones de disparo y un joystick analógico.

ILUSTRACIÓN 5. NUNCHUK

A pesar de que su creación fue para el uso en la videoconsola Wii, también se ha comenzado a utilizar en computadores. Su uso en este ámbito, incluye el control de robots (Lapping , Jenkins, Grollman, Schwertfeger, & Hinckle), aplicaciones de realidad aumentada, medicina (Gallo, De Pietro, & Marra, 2008) y otras interacciones con el propio ordenador.

2.1.2.2. PlayStation Move

PlayStation Move es un sistema de control de videojuegos para la consola PlayStation 3 de Sony. Para la utilización de este periférico, son necesarios los siguientes componentes que forman parte de PS Move: mando move, control de navegación y PlayStation Eye.

El mando move está compuesto por un giroscopio que permite establecer el ángulo en el que está posicionado el mando, así como acelerómetros que miden la velocidad de reacción y un sensor de campo magnético que mide el viraje horizontal. Además, dispone de una esfera que se ilumina con la que podremos situarnos en la habitación y que será la referencia del PlayStation Eye, y los botones clásicos del mando de PlayStation. Tiene conexión Bluetooth 2.0 y USB 2.0.

Por su parte, el control de navegación dispone también de los botones clásicos de mando de PlayStation, además de un joystick analógico. Al igual que en el caso del mando, la conexión se realiza mediante bluetooth 2.0 y USB 2.0.

ILUSTRACIÓN 6. MANDO MOVE + CONTROL DE NAVEGACIÓN

En cuanto al PlayStation Eye, se trata de una cámara RGB que toma imágenes a 120 frames por segundo a una resolución de 320x240 o a 60 frames por segundo en caso de que la resolución sea 640x480. Además, dispone de un micrófono capaz de reducir el ruido de fondo, prestando así más atención a la voz. Su entrada de audio tiene 4 canales de 16 bits cada uno.

ILUSTRACIÓN 7. PLAYSTATION EYE

El funcionamiento del PlayStation Move es similar al del WiiMote explicado con anterioridad. De este modo, haciendo uso de los diferentes sensores que posee puedan detectarse los movimientos y rotaciones realizadas con el mando de una forma precisa. La gran diferencia entre estos dos periféricos es la bola luminosa de la que dispone el PlayStation Move, que es la encargada junto con el PlayStation Eye de posicionar el mando en el entorno del juego. Con esta combinación, la PlayStation es capaz de saber la posición exacta del mando, así como su grado de inclinación, pudiendo conocer además si se ha producido un giro sobre su eje, válido para juegos en los que puedan realizarse golpes con efecto que puedan modificar el resultado final.

2.1.2.3. Kinect

Kinect es un dispositivo que nos hará olvidarnos completamente del uso de controles para interaccionar con los juegos o el propio ordenador, simplemente utilizaremos nuestro propio cuerpo para comunicarnos y mostrarle las acciones que se deben realizar.

Kinect dispone de una cámara RGB y dos sensores 3D de profundidad que nos proporcionarán una experiencia totalmente diferente. La combinación de estos sensores hace que podamos ver las imágenes capturadas en tres dimensiones, es decir, en los ejes x-y-z, pudiendo así conocer la distancia a la que se encuentra cada uno de los objetos de la imagen. Además su software interno hace posible el reconocimiento de esqueletos mediante un patrón que nos dice que un cuerpo está formado por una cabeza, un tronco, dos brazos y dos piernas, de forma que separa el esqueleto del resto de la imagen obtenida con el sensor.

KINECT

ILUSTRACIÓN 8. KINECT

Además, el sensor Kinect dispone de un array de 4 micrófonos que nos permitirán disponer de un reconocimiento de voz para dar órdenes a nuestros juegos o aplicaciones. La disposición de estos micrófonos nos ayudará a saber de dónde proceden los sonidos. El sensor dispone también de eliminación de ruido y eco, con lo que el tratamiento de los sonidos se hará más preciso.

En capítulos posteriores se profundizará tanto en el hardware como en funcionamiento de este dispositivo.

2.1.3. Justificación

A continuación realizaremos un pequeño análisis comparativo entre las tres opciones más actuales (WiiMote, PS Move y Kinect), de modo que podamos llegar a una conclusión de cuál de ellos se adecúa más a las necesidades del proyecto y por lo tanto, justificar nuestra decisión.

			
Mando	SI	SI	NO
Cámara	NO	SI	SI
Resolución	-	640x480 a 60 fps 320x240 a 120 fps	640x480 a 30 fps
Detecta movimientos	SI	SI	SI
Reconocimiento cuerpo	NO	NO	SI
Reconocimiento de voz	NO	SI	SI
Soporte para PC	SI	SI	SI
Precio[1]	51,90 €	56, 90 €	149,90 €

TABLA 1. COMPARATIVA WIIMOTE VS. PS MOVE VS. KINECT

Aunque los tres dispositivos podrían ser totalmente aptos para poder realizar un control preciso de una aplicación educativa, Kinect nos ofrece muchas más posibilidades. En primer lugar, Kinect reconoce el cuerpo entero mientras que los demás dispositivos están sujetos al movimiento del brazo con el que se sujeta el mando, o ambos brazos (para lo que sería necesario disponer de varios mandos). De este modo, con Kinect podemos realizar un mayor número de movimientos, ya que reconoce hasta 20 partes del cuerpo diferentes (Joints). Otro de los puntos fuertes de Kinect, es que podemos olvidarnos por completo de tener un mando en la mano, de modo que simplemente con un gesto de nuestro cuerpo podamos realizar acciones en nuestro pc.

El uso de Kinect en ordenador está mucho más extendido que en el resto de casos, pudiendo encontrar muchos más ejemplos y ámbitos de uso que nos facilitarán las cosas a la hora de desarrollar una aplicación para este dispositivo.

[1] Precio relativo únicamente al propio dispositivo

2.2. LIBRERÍAS

En este punto describiremos brevemente las dos principales librerías existentes para el desarrollo de aplicaciones para el sensor Kinect. Por un lado, encontramos la versión oficial de Microsoft en su versión 1.0. Por otro lado, disponemos de una versión open source (OpenNI), con la que también se podrá comenzar a desarrollar aplicaciones. Finalmente, se procederá a realizar una comparación entre ambas.

2.2.1. Microsoft SDK Kinect v1.0

El SDK de Microsoft, es el SDK oficial sacado por la compañía para la creación de aplicaciones para Kinect. Su uso está orientado principalmente a la investigación académica y a programadores particulares.

Este SDK solo es utilizable en entornos Windows 7 y los programas deben crearse en los lenguajes de programación permitidos por el SDK, en este caso C++, C# y Visual Basic, por lo que resulta algo restrictivo en ese aspecto.

ILUSTRACIÓN 9. MICROSOFT KINECT

El SDK incluye los drivers para el uso de Kinect en un ordenador, interfaces de los dispositivos con documentación para los desarrolladores y ejemplos de código fuente. Además, con él podremos realizar seguimiento del esqueleto de una o dos personas que estén en el ángulo de visión de Kinect, calcular la distancia de los objetos gracias a la cámara de profundidad o procesar audio gracias a los cuatro micrófonos que posee el sensor, entre otras cosas.

2.2.2. OpenNI

OpenNI proporciona una API para escribir aplicaciones que utilizan la interacción natural (NI, *Natural Interaction*). Esta API cubre la comunicación entre dispositivos de bajo nivel (como sensores de visión y audio) y dispositivos de alto nivel (como el seguimiento visual utilizando la visión por ordenador).

OpenNI se escribe y distribuye bajo la GNU LGPL (*Lesser General Public License*) lo que significa que su código fuente es libremente distribuido y disponible para el público en general.

ILUSTRACIÓN 10. OPENNI

OpenNI es un framework multi-idioma y multiplataforma que define las API para escribir aplicaciones que utilizan interacción natural. El propósito principal de OpenNI es formar una API estándar que permite la comunicación con:

- Sensores de audio y video (dispositivos que "ven" y "oyen" las figuras y sus alrededores).
- Middleware de visión y percepción de audio (el software que analiza el audio y los datos visuales que son registrados y los comprende).

OpenNI es un SDK no oficial de código abierto que puede utilizarse para la creación de aplicaciones para Kinect, así como para otros dispositivos similares. OpenNI fue creado en noviembre de 2010, y detrás de ella se encuentra la empresa PrimeSense que fue una de las que desarrolló la tecnología usada en Kinect. Las APIs de OpenNI ofrecen soporte para:

- Análisis del cuerpo completo.
- Reconocimiento de gestos de las manos
- Reconocimiento de voz

Cabe destacar que OpenNI permite el desarrollo de aplicaciones para Kinect en multitud de lenguajes de programación.

2.2.3. Justificación

A continuación realizaremos un pequeño análisis comparativo de las dos librerías que hemos visto para el desarrollo de aplicaciones para Kinect. De esta forma veremos de forma muy práctica y visual los puntos fuertes y débiles de cada una de ellas y justificaremos el porqué del uso de unas de ellas para llevar a cabo nuestro proyecto. Podemos ver el análisis en la siguiente tabla: (Hinchman, 2011)

	Microsoft	OpenNI
Skeletal Tracking	SI	SI
Joints	20 Joints	20 Joints
Retardo	Milisegundos	Milisegundos
Necesita calibración	NO	SI
Seguimiento Predictivo	SI	NO
Resolución cámara RGB	1024x768	800x600
Grabación de audio	SI	NO

	Microsoft	OpenNI
Cancelación de ruido	SI	NO
Reconocimiento de voz	SI	NO
Plataformas	Solo Windows 7	Windows, OS X, Linux
Licencia	Solo uso comercial	Incluye uso comercial
Múltiples sensores	SI	Si, aunque instalación compleja
Instalación	Sencilla, un único archivo	Tres instaladores separados
Seguimiento de las manos	NO	SI
Consumo	Consume más CPU	Consume menos CPU
Eventos cuando un usuario entra o sale de la escena	NO	SI
Lenguajes soportados	C++, C#, Visual Basic utilizando Visual Studio	Python, C, C++, C#, Java, Lisp y más
Documentación	Bien documentada, posee un foro de soporte	No existe foro oficial aunque existe lista de email, twitter e IRC

TABLA 2. COMPARATIVA MICROSOFT SDK VS OPENNI

Como podemos observar, ambas opciones tienen sus pros y sus contras. A pesar de las restricciones en cuanto a plataformas y lenguajes de programación para el desarrollo de aplicaciones, decidimos optar por el SDK oficial de Microsoft. El principal motivo de esta elección reside en el hecho de tener una documentación consistente, es decir, disponemos de un lugar específico en el que poder consultar nuestras dudas a la hora del desarrollo. Además, esta documentación nos permite comenzar a programar en un breve espacio de tiempo debido a su sencillez y su rápido entendimiento. Cabe destacar también que la mayor resolución de la grabación de la cámara nos permite que nuestras aplicaciones sean aún más precisas, lo que es un punto a favor para el SDK de Microsoft. Por último, el reconocimiento de voz puede sernos muy útil para proyectos futuros, ya que actualmente no se ha contemplado esa posibilidad.

3. MICROSOFT KINECT

Kinect es un dispositivo de control de movimiento que fue inicialmente creado para jugar con la videoconsola XBOX 360 sin la necesidad de ningún tipo de mando o controlador. De este modo, el jugador hace uso de su propio esqueleto para interactuar con el juego, creando una experiencia de usuario más realista y dando la opción al jugador de ser el protagonista de la historia.

Aunque el sensor Kinect ha sido desarrollado oficialmente por Microsoft, el diseño y la tecnología fue creada por la empresa israelí PrimeSense. Kinect fue anunciado por primera vez el 1 de Junio de 2009 en la Electronic Entertainment Expo 2009 bajo el nombre de "Project Natal" y posteriormente, en noviembre de 2010 sale al mercado como accesorio de la XBOX 360.

Con el lanzamiento de Kinect surgió una gran oportunidad para científicos, aficionados e inventores que comenzaron a utilizar y "piratear" Kinect para construir nuevas aplicaciones y conseguir aprovechar el potencial del que dispone este periférico. Tanto es así, que al poco de su lanzamiento, la empresa *Adafruit* ofreció una recompensa a la primera persona que consiguiera hackear Kinect (Adafruit, 2010), siendo el ganador el español Héctor Martin. Poco más tarde, la empresa PrimeSense lanzó el primer SDK no oficial para Kinect.

La primera reacción de Microsoft fue combatir a los piratas, aunque poco después se dio cuenta de que la liberación del software supondría una gran cantidad de nuevas herramientas para los usuarios. Así, el 16 de Junio de 2011 se publicó la primera beta de su SDK oficial, compatible con Windows 7, con licencia no comercial. Esta beta fue actualizada en noviembre de ese mismo año y, finalmente, en febrero de 2012 salió a la luz la versión 1.0 del SDK oficial de Microsoft.

Con la liberación del software y la investigación realizada en entornos Windows con Kinect, se han podido realizar serios avances en multitud de disciplinas como marketing, medicina, negocios, ciencias de la computación, el entretenimiento y la robótica.

Las ventas del sensor Kinect superan actualmente los 10 millones de unidades vendidos en todo el mundo (Acebo, 2011). Además, el libro Guiness World Records ha indicado oficialmente que es el producto de consumo electrónico de más rápida venta de la historia (Guinness World Records, 2011). En dos meses, se vendieron una media de 133.333 unidades por día, haciendo así un total de 8 millones de unidades en sus primeros 60 días.

3.1. HARDWARE: PARTES FUNDAMENTALES

El sensor Kinect dispone de cuatro partes fundamentales que podemos ver en la siguiente imagen y que explicaremos a continuación:

ILUSTRACIÓN 11. HARDWARE KINECT

1. **Sensores 3D de profundidad.** Los sensores tridimensionales hacen un seguimiento del cuerpo dentro del área del juego. Dispone de un proyector de profundidad (retícula izquierda) y un sensor de profundidad (retícula derecha), que calculan la distancia en función del tiempo que tarda en reflejar la luz.
2. **Cámara RGB.** Una cámara RGB (*Red, Green, Blue*) que ayuda identificar y captar imágenes y videos con una resolución máxima de 640x480 a 30 fps.
3. **Array de micrófonos.** Se usa un conjunto de cuatro micrófonos en el borde frontal inferior del sensor Kinect para el reconocimiento de voz.
4. **Inclinación motorizada.** Se trata de un impulso mecánico en la base del sensor Kinect que inclina de manera automática el sensor hacia arriba o abajo según sea necesario hasta un máximo de 27º.

Además de estas partes que podemos observar en el propio sensor, Kinect dispone de lo siguiente:

- Memoria RAM de 512 MB.
- Acelerómetro, para estabilizar la imagen cuando se mueve.
- Ventilador, que no se encuentra encendido continuamente para no interferir con los micrófonos.

3.2. FUNCIONAMIENTO BÁSICO

El funcionamiento básico de Kinect consta de tres partes diferenciadas. Por una parte, el reconocimiento de imágenes, por otro el reconocimiento de voz y, finalmente, el motor de inclinación (Bunker, 2011). A continuación se explicará brevemente en que consiste cada uno de ellos.

3.2.1. Reconocimiento de imágenes

La configuración óptica de Kinect permite el reconocimiento de imágenes en tiempo real. La tecnología usada está disponible desde hace más de 15 años, por lo que no es altamente compleja. Microsoft, por su parte, ha conseguido algunos efectos y acciones que antes únicamente estaban disponibles a precios muy elevados.

El sensor está compuesto de dos partes principales: un proyector y una cámara de infrarrojos VGA. La profundidad de los objetos es captada por la cámara gracias al rebote de los haces laser por el campo de juego, creando así un "campo de profundidad" que permite al sensor diferenciar entre los objetos estáticos de la sala y las personas utilizándolo. Este "campo de profundidad" consiste, básicamente, en que Kinect recibe este haz de luz como infrarrojos que varían en mayor o menor grado de color dependiendo de la distancia a la que se encuentran del sistema. De este modo, los cuerpos aparecen como rojo, verde, etc.; y los objetos más alejados aparecen en gris.

Con los datos obtenidos en esta imagen, el software aplica una serie de filtros para que Kinect pueda saber qué es una persona y qué no, basándose en una serie de directrices como "una persona tiene dos brazos y dos piernas".

Una vez tenemos la información ordenada, se identifican las partes del cuerpo y crea un esqueleto en movimiento. Kinect tienes unas 200 posturas precargadas, de manera que se puedan llenar los espacios en blanco en caso de que se realicen movimientos que obstruyan la visión de su esqueleto. El principal inconveniente que podemos encontrar, es que los dedos no se asignan de forma individual en el esqueleto, impidiendo con ello una serie de movimientos.

Todo esto es realizado por el sistema continuamente a 30 fps.

3.2.2. Reconocimiento de voz

El principal problema del reconocimiento de voz en Kinect, es que tiene que ser sensible a las voces hasta 5 metros de distancia, además de que debe ser capaz de ignorar los ruidos ambientales y cualquier otro sonido de la propia voz. Para poder realizar esto, el equipo de Microsoft realizó una serie de pruebas en diferentes viviendas con 16 micrófonos. En estas pruebas se tomaron una serie de grabaciones de diferentes configuraciones, de forma que pudieran conseguir el mejor posicionamiento de los micrófonos en el sensor.

El resultado de estas pruebas proporcionó la configuración actual de los micrófonos en Kinect. Se trata de un array de cuatro micrófonos posicionados boca abajo, uno a la izquierda y tres en la parte derecha.

Este conjunto de micrófonos recoge las voces de la mejor manera posible. Aun así, necesita ayuda del software de la cámara. La unidad de procesamiento cancela el ruido, mientras que un software usa la cámara para calcular de donde proviene el sonido, de modo que crea una burbuja alrededor del usuario y consigue separar la voz del usuario, omitiendo la del resto de personas que se encuentren alrededor.

3.2.3. Motor de inclinación

Tras las investigaciones de Microsoft en diferentes entornos y distintas configuraciones de los espacios y dadas las diferentes posibilidades que pueden encontrarse, Microsoft decidió que lo óptimo sería que la cámara pudiera moverse arriba y debajo de forma que pudiera adaptarse a cualquier entorno y cualquier campo de visión, calibrándose al espacio concreto y no teniendo que adaptar el espacio a la cámara.

El motor se encuentra en la base de la cámara, pudiendo moverse arriba y abajo un total de 27º. La altura recomendada de utilización de la cámara para que el espacio de visión sea adecuado es de entre 1 y 2 metros. Además, existe un ventilador que únicamente entra en acción si es necesario, de forma que no interfiera con los micrófonos.

3.3. ESPECIFICACIONES DEL SENSOR KINECT

A continuación se mostrarán las especificaciones técnicas del sensor Kinect, de modo que podamos conocer más sobre sus características y sobre cómo utilizarlo.

Especificaciones del sensor Kinect	
Ángulo de visión	43º de campo de visión vertical 57º de campo de visión horizontal
Ángulo de movimiento del motor	±27º
Velocidad de frames	30 fps
Resolución (profundidad)	VGA (640x480)
Resolución (cámara RGB)	VGA (640x480)
Formato de audio	16 KHz, 16 bit PCM
Características de entrada de audio	Array de cuatro micrófonos con un convertidor analógico a digital de 24 bits (ADC) y procesamiento de señales de Kinect con cancelación de eco y eliminación de ruido de fondo.

TABLA 3. ESPECIFICACIONES DEL SENSOR KINECT

3.4. RANGOS DE UTILIZACIÓN DEL SENSOR KINECT

El sensor Kinect está diseñado para videojuegos y escenarios similares. Originalmente estaba diseñado para su uso en XBOX 360, aunque con el SDK de Windows y el nuevo sensor Kinect para Windows, puede ser utilizado en otros contextos. A pesar de este cambio, tanto el diseño físico como las capacidades del sensor no se ven modificadas.

En la siguiente imagen podemos ver los rangos aplicados tanto al sensor Kinect de XBOX 360 (*Default Range*), así como al sensor Kinect para Windows (*Default Range* y *Near Range*).

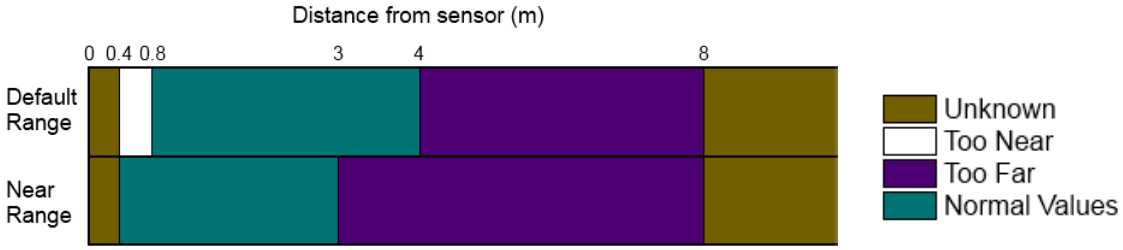


ILUSTRACIÓN 12. RANGOS DE UTILIZACIÓN DEL SENSOR KINECT

Este esquema se aplica a los valores devueltos por la API administrada, o por la API nativa con el flag NUI_IMAGE_STREAM_FLAG_DISTINCT_OVERFLOW encendido. Con este flag apagado, todos los rangos excepto para el caso de “valores normales” devuelve cero.

Con esta imagen podemos observar como la distancia óptima del sensor, para poder utilizarlo sin ningún tipo de problema, se encuentra entre 1 y 4 metros. Esta distancia es ligeramente inferior (entre 0.5 y 3 metros) en el caso del *Near Mode* del sensor Kinect para Windows.

Cabe destacar, que el sensor Kinect solo detecta esqueletos que se encuentran de pie, no reconoce figuras que se encuentren sentadas.

3.5. Recomendaciones de Uso

A continuación enumeraremos una serie de recomendaciones para un uso óptimo del sensor Kinect, así como para evitar desperfecto en el aparato.

- Coloque el sensor sobre una superficie estable, en un lugar en el que no pueda caerse o ser golpeado durante su uso.
- Ajuste la ubicación del sensor Kinect sólo moviendo la base.
- No ajuste el ángulo de visión del sensor con la mano, moviendo el motor de inclinación manualmente. Una vez instalado, los motores del sensor ajustaran el ángulo de visión para evitar el riesgo de dañar el sensor de Kinect.
- No coloque el sensor Kinect delante o sobre un altavoz o superficie que vibre o haga ruido.
- Mantenga el sensor Kinect fuera del alcance de la luz solar directa.
- No utilice el sensor Kinect cerca de fuentes de calor.
- Utilice el sensor Kinect dentro de su rango de temperatura operativa (entre 5 y 35 grados centígrados).
- Si el sensor está expuesto a una temperatura superior a su rango determinado, apáguelo y deje que la temperatura se estabilice antes de usarlo de nuevo.
- El sensor Kinect está protegido contra el sobrecalentamiento mediante un ventilador. Este ventilador está controlado por el firmware del sensor y apagará la cámara si la temperatura llega a los 90 grados centígrados. No existe una interfaz de software para controlar el ventilador del dispositivo.
- Es necesario conectar el dispositivo a una fuente de alimentación externa, ya que, en caso contrario, su funcionalidad será limitada.
- Si dispone de varios sensores Kinect, asegúrese de que están conectados en diferentes hub USB.
- Asegúrese de que el dispositivo no está conectado en un hub utilizado por otros dispositivos.
- Los controladores de Kinect no se instalarán correctamente, y no funcionarán correctamente su existe otro controlador para el sensor instalado en el PC.
- No requiere herramientas para la calibración de audio y video.

4. KINECT FOR WINDOWS SDK V1.0

Poco después del lanzamiento de Kinect, la comunidad de investigadores y desarrolladores de software, tanto a nivel profesional como a nivel particular, comenzaron a hackear el sensor. Esto provocó la aparición de las primeras distribuciones libres para el desarrollo de nuevas aplicaciones para pc, como por ejemplo OpenNI de PrimeSense. Visto el éxito obtenido, Microsoft decidió publicar un SDK oficial de Kinect.

Este SDK está orientado a la investigación académica y a programadores particulares con el objetivo de experimentar en la creación de nuevas interfaces naturales de usuario. A pesar de esta licencia no comercial, se espera que en el futuro Microsoft publique un SDK con vistas comerciales.

A continuación comentaremos brevemente el contenido del SDK, así como algunas de las cosas que nos permite hacer:

El SDK incluye:
Drivers y documentación técnica para la implementación de aplicaciones utilizando el sensor Kinect.
APIs de referencia y documentación para la programación. La API ofrece flujos de múltiples medios con una latencia mínima.
Ejemplos que muestran las buenas prácticas para el uso de un sensor Kinect, así como un navegador de ejemplo para la navegación, instalación y ejecución de las muestras.
How tos que descomponen las muestras en tareas para el usuario.

TABLA 4. ELEMENTOS QUE INCLUYE EL SDK DE KINECT

A continuación veremos unos ejemplos de las características que se pueden implementar con el SDK de Kinect para Windows:

El SDK permite
Reconocer y realizar un seguimiento de una o dos personas que se desplazan dentro del campo de visión del sensor mediante *skeletal tracking.*
Determinar la distancia de un objeto desde el sensor utilizando una cámara de profundidad XYZ que tenga acceso al flujo de color además de una corriente de profundidad.
Captura de audio con ruido y cancelación de eco o encontrar la ubicación de la fuente de audio utilizando un haz durante la captura de audio con un array de cuatro micrófonos.
Reconocer el habla con la captura de audio integrado en el Microsoft Speech Recognition API

TABLA 5. CARACTERÍSTICAS QUE SE PUEDEN IMPLEMENTAR CON EL SDK DE KINECT

4.1. NOVEDADES EN LA VERSIÓN V1.0 DEL SDK

Como se comentó anteriormente, las primeras versiones del SDK de Kinect fueron versiones Beta. Finalmente, en febrero de 2012 se publicó la versión v1.0 del SDK que incorporaba algunas novedades respecto a la beta anterior. A continuación, mostraremos algunas de estas novedades y diferencias más importantes entre el SDK Beta 2 y el SDK v1.0.

- Soporte para hasta 4 sensores Kinect conectados en el mismo equipo, asumiendo que el equipo es lo suficientemente potente y que se conectan a diferentes controladores USB por lo que hay suficiente ancho de banda disponible. El *skeletal tracking* solo se puede realizar en un sensor Kinect por proceso. El programador será quien decida qué sensor Kinect utilizar.
- Está disponible la función *Near Mode*, aunque solo es funcional para el sensor Kinect para Windows, no para el dispositivo de XBOX 360.
- Mejora la robustez, incluyendo la estabilidad del driver, el tiempo de ejecución y correcciones del audio.
- Actualizaciones y mejoras de la API.
- Mejora en el manejo de errores.
- FPS corregidos para el modo de alta resolución.
- El SDK de Kinect ahora incluye los componentes más recientes de Microsoft Speech.
- Modelo acústico actualizado que mejora la precisión de los valores devueltos por la Speech API.
- Se ha añadido un explorador de muestra, de manera que resulte más sencillo encontrar y ver los ejemplos.

4.2. REQUERIMIENTOS DEL SISTEMA

A continuación mostraremos los requisitos del sistema necesarios para el desarrollo de aplicaciones utilizando el SDK de Kinect para Windows. Trataremos tanto los sistemas operativos soportados como los requerimientos hardware y software, así como los prerrequisitos para desarrolladores.

4.2.1. Sistemas Operativos y Arquitecturas soportadas

Se debe ejecutar la aplicación creada utilizando el SDK en un entorno nativo de Windows. No se podrá ejecutar aplicaciones en una máquina virtual, porque los drivers de Kinect y su SDK deben estar instalados en el equipo que está ejecutando la aplicación. Los sistemas operativos soportados son:

Sistemas Operativos y Arquitecturas soportadas
Windows 7
Windows 7 Embedded Standard 7

TABLA 6. SISTEMAS OPERATIVOS Y ARQUITECTURAS SOPORTADAS

4.2.2. Requerimientos de Hardware

Para poder ejecutar y desarrollar aplicaciones para Kinect, debemos disponer del siguiente hardware:

Requerimientos de Hardware
Sensor Kinect
Ordenador con las siguientes características hardware: • Procesador de 32 bits (x86) o 64 bits (x64). • Procesador dual-core, 2.66 GHz o superior. • Bus USB 2.0 dedicado para el sensor Kinect. • 2 GB de memoria RAM. • Tarjeta gráfica que soporte DirectX 9.0c

TABLA 7. REQUERIMIENTOS HARDWARE

4.2.3. Requerimientos de Software

Para el correcto desarrollo y ejecuciones de las aplicaciones realizadas para Kinect se debe disponer del siguiente software:

Requerimientos de Software
Microsoft Visual Studio 2010 Express (o cualquier otra edición)
.NET Framework 4 (instalado con Visual Studio 2010)
Microsoft Speech Platform – Runtime (Version 11) (Instalada automáticamente como parte del Kinect Runtime Setup)
Microsoft Speech Platform – Software Development Kit (SDK Version 11)

TABLA 8. REQUERIMIENTOS DE SOFTWARE

4.2.4. Prerrequisitos para desarrolladores

Para aprovechar las características del SDK, los desarrolladores deberán estar familiarizados con entornos de desarrollo (como Visual Studio) y lenguajes de programación (como C# y C++).

4.2.4.1. Visual Studio.

ILUSTRACIÓN 13. VISUAL STUDIO

Visual Studio (Microsoft) es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE) que habilita el uso compartido de herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML.

El uso de Visual Studio nos permite la incorporación de numerosas tecnologías como .NET Framework, WPF, Silverlight, Windows Forms o ASP.NET entre otros (Microsoft).

Además de la variedad de tecnologías que podemos incorporar si utilizamos Visual Studio, éste también nos proporciona muchas plantillas de aplicación y nos ayuda a crear programas en multitud de lenguajes de programación. En cuanto a los lenguajes de programación que utiliza se incluyen Visual Basic, Visual C# y Visual C++. Por su parte, nos permite la realización de aplicaciones para Windows, aplicaciones web, aplicaciones Office o aplicaciones de SharePoint.

Aunque, existen diferentes versiones de licencia propietaria, como *Visual Studio 2010 Ultimate, Visual Studio 2010 Premium* o *Visual Studio 2010 Professional*, desde la versión 2005 Microsoft ofrece gratuitamente las *Express Editions*, que son ediciones básicas separadas por lenguajes de programación.

4.2.4.2. C++

(Deitel H. , 1994) C++ es una ampliación de C que fue desarrollado por Stroustrup en los Laboratorios Bell. C++ permite llevar a cabo programación orientada a objetos. Los objetos son componentes de software reutilizables que modelan elementos del mundo real. El uso de objetos proporciona una manera rápida y correcta de elaboración de software. El desarrollo orientado a objetos hace que los grupos de desarrollo de software sean entre 10 y 100 veces más productivos que lo que era posible anteriormente con técnicas convencionales de programación.

El lenguaje C++ se comenzó a desarrollar en 1980. Al comienzo fue una extensión del lenguaje C que fue denominada *C with clases*. Ante la gran difusión y éxito obtenido en el mundo de los programadores, se comenzó a estandarizar, hasta que en 1989 se formó un comité ANSI para estandarizarlo a nivel americano e internacional.

En la actualidad C++ es un lenguaje versátil, potente y general, siendo de este modo una herramienta muy utilizada para el desarrollo de aplicaciones. C++ mantiene las ventajas de C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, eliminó algunas de las dificultades y limitaciones del C original.

C++ es a la vez un lenguaje procedural y orientado a objetos. Como lenguaje procedural se asemeja a C y es compatible con él aunque presenta ciertas ventajas. Como lenguaje orientado a objetos se basa en una filosofía completamente diferente, que exige un completo cambio de mentalidad del programador.

4.2.4.3. C#

(Deitel H. M., 2007) Debido al gran avance de los dispositivos electrónicos (como teléfonos móviles o PDA's), han surgido problemas y nuevos requisitos de programación. Uno de los problemas principales, consistía en la dificultad de la integración e instalación de software de diversos lenguajes. Además, por este avance, los desarrolladores se dieron cuenta de que debían ampliar sus horizontes, de modo que las aplicaciones no fueran únicamente accesibles desde un ordenador persona, sino que también fueran accesibles a través de la web o de los dispositivos móviles. Para satisfacer estas necesidades, en el año 2000, Microsoft anunció el lenguaje de programación C#. Este lenguaje fue desarrollado en Microsoft por un equipo dirigido por Anders Helsberg y Scott Wiltamuth y fue diseñado específicamente para la plataforma .NET, como un lenguaje que permitiera a los programadores migrar fácilmente hacia .NET. Tiene sus raíces en C, C++ y Java, adaptando las mejores características de estos lenguajes y agregando características propias. C# es un lenguaje de programación orientado a objetos y contiene una poderosa biblioteca de clases, que consta de componentes preconstruidos que permiten a los programadores desarrollar aplicaciones con rapidez.

C# es un lenguaje de programación visual, controlado por eventos, en el cual se crean programas mediante el uso de un Entorno de Desarrollo Integrado (IDE). Mediante este IDE, el programador podrá crear, ejecutar, probar y depurar programas en C#. Gracias a la plataforma .NET, los lenguajes serán interoperables, de modo que los componentes software de distintos lenguajes pueden interactuar como nunca antes se había hecho. Otro punto importante, es que las aplicaciones en C# pueden interactuar a través de Internet mediante el uso de estándares como XML y el Protocolo Simple de Acceso a Objetos (SOAP).

El lenguaje de programación C# original se estandarizó a través de Ecma Internacional (ECMA International) en diciembre de 2002, como *Estándar ECMA-334: Especificación del lenguaje C#* (ECMA International).

4.2.4.4. Microsoft .NET

(Deitel H. M., 2007) En el año 2000, Microsoft anunció su iniciativa .NET (Microsoft), a través de la cual se podría incluir Internet y Web en el desarrollo y uso de software. El hecho de que .NET sea independiente del lenguaje o plataforma hace de este framework algo muy útil. De este modo, los desarrolladores pueden crear y contribuir en aplicaciones sin estar forzados a utilizar un lenguaje concreto, simplemente utilizando uno compatible con .NET.

La arquitectura .NET puede existir en diferentes plataformas, no solo en sistemas basados en Microsoft Windows, lo que hace a los programas más portables. Un componente clave de la arquitectura .NET son los Servicios Web. Los Servicios Web son componentes de software reutilizables que pueden ser utilizados a través de Internet.

La estrategia .NET extiende el concepto de reutilización de software hasta Internet, de modo que permite a las empresas concentrarse en sus especialidades sin la necesidad de tener que implementar una solución para cada aplicación.

El .NET Framework de Microsoft es el centro de la estrategia .NET. Este marco de trabajo administra y ejecuta aplicaciones y Servicios Web, contiene biblioteca de clases (denominada FCL, *Framework Class Library*), impone la seguridad y proporciona muchas otras herramientas de programación. Los detalles de .NET Framework se encuentran en la Infraestructura de Lenguaje Común (CLI), que contiene información acerca del almacenamiento de los tipos de datos, los objetos y demás.

El *Common Language Runtime* (CLR) es otra parte central de .NET Framework. Con él, se ejecutarán los programas de .NET. Los programas se compilan en instrucciones específicas para las máquinas en dos pasos. Primero, el programa se compila en *Lenguaje Intermedio de Microsoft* (MSIL), que define las instrucciones para el CLR. El CLR puede unir código convertido de otros lenguajes o fuentes en el MSIL. Este MSIL se coloca en el archivo ejecutable de la aplicación (proceso que se conoce como ensamblaje). Cuando la aplicación se ejecuta, otro compilador (compilador JIT) en el CLR traduce el MSIL del archivo ejecutable en código máquina para una plataforma específica, pudiendo luego ejecutarlo en dicha plataforma.

4.3. INSTALACIÓN DEL SDK DE KINECT

A continuación se mostrarán los pasos para la instalación del SDK de Kinect, así como recomendaciones para que el SDK se instale de forma correcta.

1. En primer lugar se deberá descargar el archivo de instalación de la siguiente página web: http://www.microsoft.com/en-us/download/details.aspx?id=28782.
2. Asegúrese de que el sensor Kinect no está conectado a cualquiera de los puertos USB del ordenador.
3. Si existe una versión anterior del SDK de Kinect para Windows instalado en su equipo, deberá desinstalarlo antes de proceder.
4. Desinstale el resto de drivers del sensor Kinect que puedan existir en el computador.
5. Cierre Visual Studio. Deberá cerrar Visual Studio antes de instalar el SDK y reiniciarlo tras la instalación para que pueda recoger las variables de entorno que requiere el SDK.
6. Desde la ubicación de descarga del archivo mencionado anteriormente, haga doble clic en *KinectSDK-v1.0-Setup.exe*. Este instalador solo funciona para Windows, tanto de 32 bits, como de 64 bits.
7. Una vez que el SDK ha completado la instalación con éxito, deberá asegurarse de que el sensor Kinect está conectado a una fuente de alimentación externa y luego conectar el sensor Kinect en el puerto USB del PC. Los controladores se cargarán de forma automática.
8. Ahora el sensor Kinect debería funcionar de forma correcta. Para comprobarlo lanzaremos el explorador de muestras de Kinect (*Inicio → Todos los programas → Microsoft Kinect SDK v1.0 → Kinect SDK Sample Browser*) y verifique mediante estos ejemplos que funcionan todos los componentes del sensor Kinect: Skeletal tracking, el color, la imagen de profundidad y el audio. Deberá aparecer encendido un LED verde en la parte frontal del sensor Kinect.

4.4. ARQUITECTURA

El SDK proporciona una sofisticada biblioteca de software y herramientas para ayudar a los desarrolladores a utilizar la rica Interfaz Natural de Usuario (NUI, *Natural User Interface*), que reacciona ante eventos de los sentidos del mundo real.

En la siguiente figura podemos observar la interacción básica entre el sensor Kinect, la biblioteca de software y la aplicación.

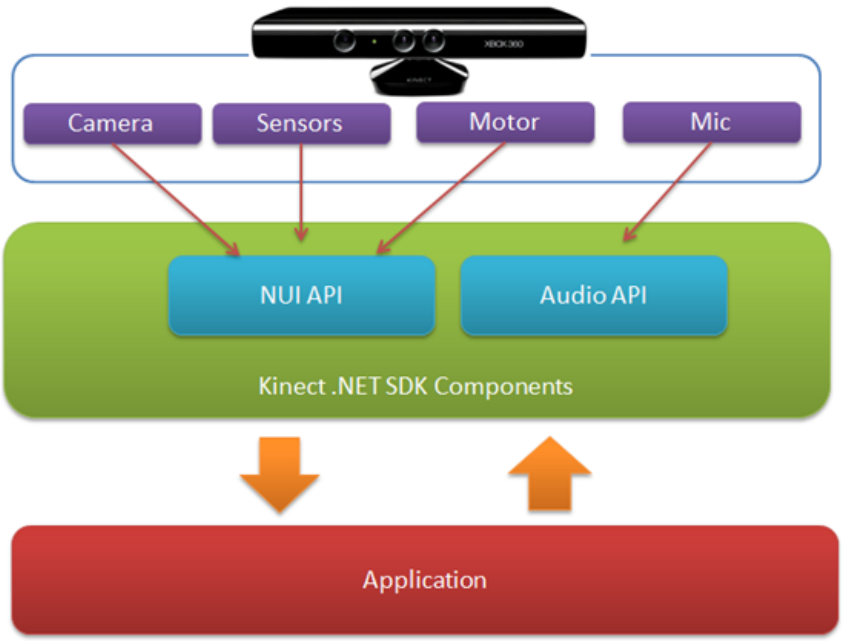


ILUSTRACIÓN 14. INTERACCIÓN DEL SENSOR, EL SOFTWARE Y LAS APLICACIONES

En este esquema podemos diferenciar cuatro partes fundamentales:

- **Hardware Kinect.** Incluyen los componentes hardware vistos anteriormente (sensores de profundidad, cámara RGB, array de micrófonos y el motor de inclinación), así como el cable USB por el que se conecta el sensor al PC.
- **Drivers de Kinect.** Los controladores de Windows para el sensor Kinect se instalan como parte del proceso de instalación del SDK, tal y como se especificó en el punto anterior. Estos controladores proporcionan soporte para:
  - El sistema de micrófonos como un dispositivo kernel-mode al que podremos acceder con las APIs de audio estándares de Windows.
  - Streaming de imagen y datos de profundidad.
  - Funciones de enumeración de dispositivos que permiten a una aplicación utilizar más de un sensor Kinect conectador al mismo equipo.
- **KinectAudio DirectX Media Object (DMO).** El KinectAudio DMO amplía el soporte de micrófonos de Windows 7 para exponer la localización de la formación de la fuente.
- **API Windows 7 estándar.** El sonido, el habla y otros medios de las API de Windows 7, tal y como se describen en el SDK de Windows 7 el SDK de Microsoft Speech.

En la siguiente imagen podemos ver todos estos componentes de forma más específica. Lo podemos dividir en cuatro grandes categorías: hardware, modo kernel, modo usuario y aplicación.

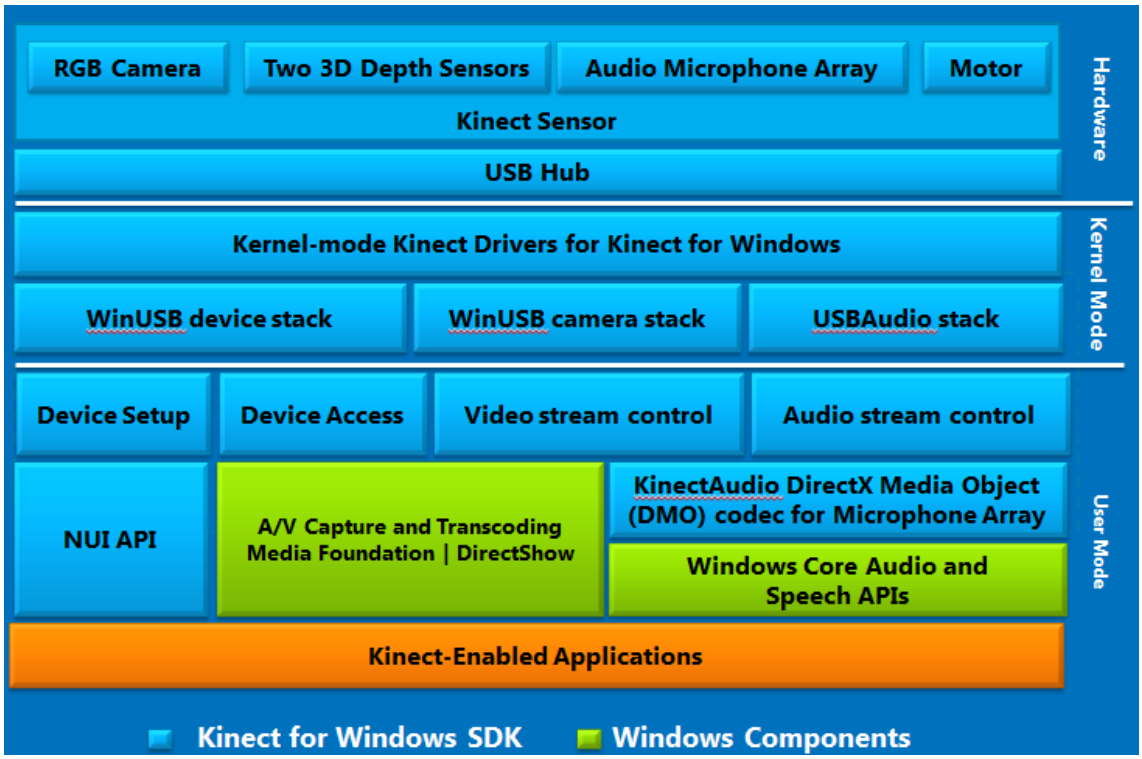


ILUSTRACIÓN 15. ARQUITECTURA DEL SDK DE KINECT PARA WINDOWS

4.5. NUI API

La NUI API es el núcleo de la API de Kinect para Windows. Ésta se encarga del procesado de la imagen y de la gestión del dispositivo que incluye las siguientes funciones:

- Acceso a los diferentes sensores Kinect que estén conectados al equipo.
- Acceso a los flujos de datos de imagen y profundidad captados por los sensores de Kinect.
- Procesado de imágenes y datos de profundidad para dar soporte al *skeletal tracking.*

4.5.1. Inicialización de la NUI API

Los controladores de Kinect permiten el uso de múltiples sensores Kinect en un mismo ordenador. Además, la NUI API incluye funciones para enumerar los sensores, de forma que se pueda determinar cuántos sensores Kinect hay conectados al equipo, tomar el nombre de cada uno de ellos y configurar las características de flujo de datos para cada uno de ellos de forma individual.

La NUI API dispone de cuatro subsistemas que pueden ser configurados en cada aplicación. Estos subsistemas pueden ser seleccionados a la hora de inicializar la API, pudiendo seleccionar uno o varios de ellos según nuestro objetivo. A continuación se enumerarán y describirán brevemente:

- **Color.** Activa el flujo de datos de imagen captados por el sensor.
- **Depth.** Activa el flujo de datos de profundidad captados por el sensor.
- **Depth and Player Index.** Activa la detección de datos de profundidad desde el sensor y asigna un índice de jugador al esqueleto generado. De este modo, se puede identificar a cada usuario captado por Kinect.
- **Skeletal Tracking.** Si se activa, la aplicación utilizará los datos de posición y seguimiento del esqueleto.

4.5.2. Visión general de los flujos de datos de imagen

La NUI API proporciona medios para modificar la configuración de Kinect, permitiéndonos acceder a los datos de imagen del sensor. El flujo de datos se entrega como una sucesión de imágenes fijas. Cuando inicializamos la NUI API podemos definir qué tipo de imágenes queremos recibir del sensor Kinect.

A continuación veremos los tipos de datos de imagen del sensor Kinect a los que tienen acceso las aplicaciones.

4.5.2.1. Datos de imagen a color

Los datos de imagen a color están disponibles en dos formatos diferentes:

- **Color RGB.** Proporciona mapas de bits de color de 32 bits lineales con formato X8R8G8B8 (que reserva 8 bits para cada color), dentro del espacio de colores sRGb. Para trabajar con este color es necesario especificar *color* o *color_YUV* cuando se abre el flujo de datos.

- **Color YUV.** Proporciona mapas de bits de 16 bits con corrección gamma. Al utilizar 16 bits por cada pixel, utiliza menos memoria y asigna menos buffer de memoria al abrir el flujo de datos. Este formato está solo disponible para una resolución de 640x480 a 15 fps.

Los dos formatos utilizan las mismas cámaras, por lo que la imagen representada será la misma en ambos casos.

El sensor Kinect utiliza una conexión USB para transferir datos al PC, aunque esta conexión tiene un ancho de banda limitado. Los datos de imagen que devuelve el sensor a 1280x1024 son comprimidos y convertidos a RGB antes de la transmisión. Una vez realizada la transmisión. Se descomprimen los datos antes de transferirlos a la aplicación. El uso de esta compresión nos permite manejar los datos de color a velocidades de hasta 30 FPS. A pesar de ello, el algoritmo que se utiliza nos lleva a una pérdida de calidad de imagen.

4.5.2.2. Datos de profundidad

El flujo de datos de profundidad nos proporciona un frame en el que cada pixel representa la distancia cartesiana entre el sensor y el objeto más cercano con su coordenadas x e y del campo de visión.

Las aplicaciones pueden procesar datos de un flujo de datos de profundidad para poder utilizar algunas características personalizadas, como el seguimiento de los movimientos del usuario o la identificación de objetos de fondo que serán ignorados durante la ejecución.

Cada pixel del flujo de profundidad utiliza 13 bits para los datos de profundidad y 3 bits para identificar al jugador. Un valor de datos de 0 indicaría que no existen datos de profundidad disponibles en esa posición, debido a que el objeto se encuentra muy cerca o muy lejos del sensor. Cuando la opción de skeletal tracking está desactivada, los 3 bits que identifican el jugador se ponen a 0.

4.5.3. Skeletal Tracking

La NUI Skeleton API proporciona información de la localización de un máximo de dos jugadores de pie enfrente del sensor Kinect. En esta información se tendrá en cuenta también la posición y orientación del cuerpo.

Estos datos son proporcionados al código como puntos, denominados *Joints*, que forman el esqueleto mediante puntos concretos, como podemos ver en la siguiente figura. Concretamente, se dispone de 20 *Joints* que se corresponde con las partes que definen nuestro cuerpo, así como con las articulaciones del mismo (cabeza, hombros, codos, muñecas, manso, cadera, rodillas, tobillos y pies). Este esqueleto representa la posición y postura actual del usuario. Para usar skeletal tracking, la aplicación debe indicarlo al inicializar la NUI y debe habilitar la opción.

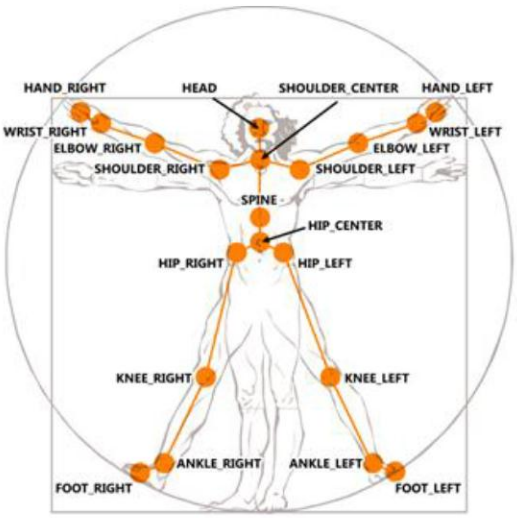


ILUSTRACIÓN 16. JOINTS DETECTADOS EN EL SKELETAL TRACKING

Como se comentó anteriormente, el Skeleton Engine permite detectar dos esqueletos completos de dos usuarios que se encuentren en el campo de visión del sensor. Esto se denomina tracking activo. Al margen de esto, también existe un tracking pasivo que reconocerá hasta 4 usuarios más, pero de los que solo almacenará la información sobre su posición. El proceso de identificar si un esqueleto es actico o pasivo se realiza de forma automática, asignándose tracking activo a los dos primeros usuarios en ser detectados.

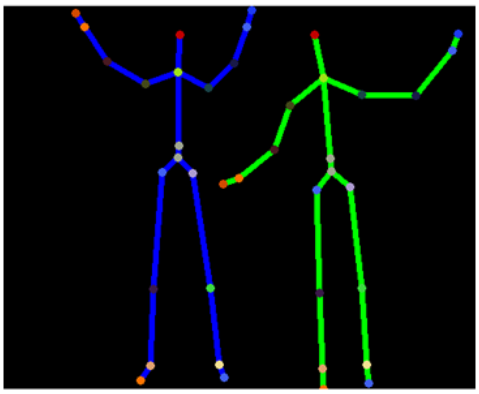


ILUSTRACIÓN 17. TRACKING ACTIVO PARA DOS JUGADORES

Se dispone de los siguientes datos para todos los esqueletos detectados:

- Estado actual del seguimiento del esqueleto asociado:
  - Para los esqueletos con tracking pasivo, se indica *position-only. Ç*
  - Para los esqueletos con tracking activo, se indica *tracked.*
- A cada usuario se le asigna un identificador único mientras se mueva por el espacio de visión del sensor. Si el usuario se sale de este campo de visión, puede que el identificador se pierda.

4.5.4. Transformaciones en la NUI

A continuación, se describirá brevemente los sistemas de coordenadas que se utilizan tanto para los datos de imagen de color y profundidad, como para el skeletal tracking.

4.5.4.1. Espacio de las Imágenes de Profundidad

Los frames de imagen del mapa de profundidades son de 640x480, 320x240, o 80x60 pixeles de tamaño. Cada uno de los pixeles representa la distancia cartesiana, en milímetros, desde el plano de la cámara al objeto más cercano mediante dos coordenadas, x e y. Si un pixel tiene valor 0, indica que el sensor Kinect no encontró ningún objeto dentro de su rango de visión.

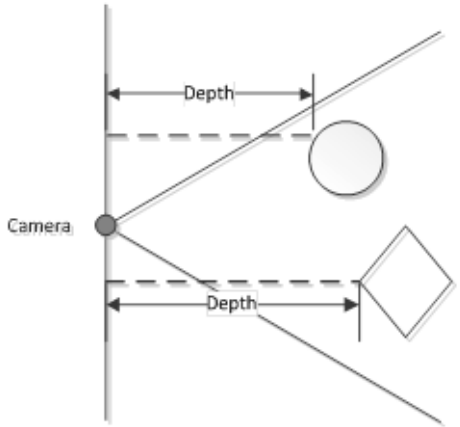


ILUSTRACIÓN 18. SISTEMA DE COORDENADAS DEL SENSOR DE PROFUNDIDAD

4.5.4.2. Espacio del Esqueleto

Las posiciones del esqueleto de los usuarios se expresan con las coordenadas x, y, z, que se expresan en metros, a diferencia del caso anterior. El sistema de coordenadas, coloca el sensor Kinect en el punto de origen, con el eje z positivo extendiéndose en la dirección a la que apunta el sensor. El eje y positivo se extiende hacia arriba, mientras que el eje x positivo lo hace hacia la izquierda (teniendo como referencia el propio sensor Kinect), tal y como se muestra en la siguiente figura.

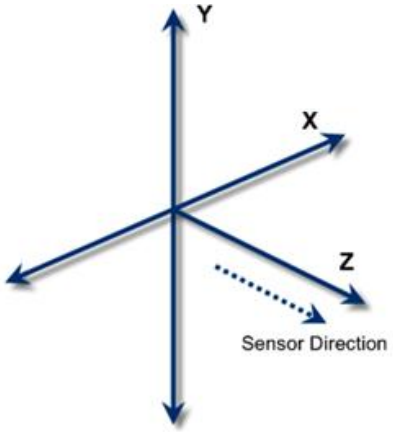


ILUSTRACIÓN 19. SISTEMA DE COORDENADAS PARA EL SENSOR KINECT

4.6. AUDIO API

El sistema de micrófonos del sensor Kinect está compuesto por un conjunto de micrófonos. Generalmente, un array de micrófonos se compone de una serie de micrófonos (normalmente cuatro) que se encuentran separados varios centímetros, siguiendo un patrón lineal o en forma de L. Un array de micrófonos de este tipo supone varias ventajas significativas respecto a un micrófono independiente:

- El uso de un array de micrófonos permite mejorar la supresión de ruido y la cancelación de eco, así como una mejora significativa en audio, ya que soporta algoritmos más sofisticados que un micrófono independiente.
- Dado que el sonido no llega a la vez a todos los micrófonos, la aplicación puede determinar la dirección del origen del audio y usar el conjunto de micrófonos como un micrófono direccional orientable.

El sensor Kinect incluye un array de cuatro micrófonos colocados de forma lineal, que utilizan ACD (*Analog to Digital* Converter) de 24 bits y proporciona procesamiento de la señal con eliminación de eco y supresión de ruido. Las aplicaciones que utilicen esta API podrán utilizar los micrófonos para lo siguiente:

- Captura de audio de alta calidad.
- Localización de la fuente de sonidos.
- Reconocimiento de voz.

5. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN

En este capítulo se detallará todo el proceso de desarrollo del proyecto, desde el análisis inicial hasta la implementación. Este será el núcleo del proyecto, ya que en él podremos ver toda la funcionalidad y se explicará en profundidad cuál es el funcionamiento del sistema completo.

Para facilitar la lectura y definir de forma clara todas las fases del desarrollo, se ha dividido este capítulo en tres partes bien diferenciadas. En primer lugar, veremos el análisis, donde detallaremos los casos de uso y requisitos del sistema. A continuación, se explicará la fase de diseño, en la que veremos cuál es la arquitectura del sistema y especificaremos cuál es el diseño de la aplicación. Por último, se verá la fase de implementación en la que veremos los aspectos más importantes para llevar a cabo la implementación de la aplicación.

5.1. ANÁLISIS

En este punto, veremos cuál es el funcionamiento básico del sistema y definiremos cuáles serán todas sus funcionalidades y restricciones a través de los casos de uso y los requisitos. Además tendremos una perspectiva global del funcionamiento de la aplicación.

Este proceso se ha llevado a cabo mediante una serie de entrevistas con el cliente (en este caso el tutor del proyecto) en las que se definían los objetivos y funcionalidades que debe cumplir la aplicación. Una vez realizado el análisis de requisitos se expondrá al cliente para su confirmación, ya que este documento servirá de contrato entre ambas partes.

5.1.1. Visión General de la Aplicación

El controlador Kinect, será el encargado de proporcionar una interacción natural entre el profesor y una aplicación educativa de realidad aumentada. La aplicación del controlador Kinect incluirá un menú de configuración que será mostrado al inicio de la aplicación (y que posteriormente podrá ser mostrado mediante un botón en la interfaz), en el que el usuario podrá decidir que gestos son los que quiere utilizar para cada una de las acciones disponibles. En el caso de no realizar ningún cambio, los gestos serán los predeterminados por la aplicación o los últimos guardados si ya se había realizado un cambio previo.

Una vez seleccionados los gestos a utilizar se mostrará una pantalla con la imagen capturada por el sensor Kinect. En este punto, podrán reconocerse los gestos del usuario para interactuar con la aplicación. Cuando se encuentre una coincidencia, es decir, cuando se haya reconocido un gesto del usuario, se llevará a cabo la acción correspondiente asignada en el menú de configuración antes mencionado. Las acciones a realizar pueden ser de dos tipos. Por un lado, tenemos las acciones que realizan control de las diapositivas de un PowerPoint, en cuyo caso, nuestra aplicación será la encargada de ir a la diapositiva anterior o siguiente según el gesto reconocido. Por otro lado, tenemos las acciones propias de la aplicación educativa de realidad aumentada. En este último caso, comunicaremos mediante un socket a la aplicación educativa un dato numérico con el que interpretará la acción que tiene que llevar a cabo.

El sensor Kinect está continuamente leyendo las posiciones de las distintas partes del cuerpo, de modo que, para evitar que se reconozca el gesto durante todo el periodo de tiempo que mantengamos la posición, se ha decidido que no podrá reconocerse un nuevo gesto hasta que el anterior haya sido deshecho, es decir, hasta que se haya vuelto a una posición que no genere una coincidencia.

El perfil de usuario que utilizara la aplicación será un profesor en el entorno de una clase. A pesar de este perfil, el usuario no tiene por qué tener conocimientos muy avanzados sobre la tecnología y su funcionamiento, por lo que la aplicación deberá ser intuitiva y fácil de utilizar para que el profesor pueda seguir con la dinámica de la clase sin ningún tipo de problema o retraso.

5.1.2. Casos de Uso

Los casos de uso nos permitirán describir qué hace el sistema desde el punto de vista del usuario, es decir, nos detallarán el uso del sistema y cómo éste interactúa con los usuarios. De este modo se podrá observar las interacciones típicas en un usuario (actor) y el sistema, describiendo qué se hace sin entra en el cómo lo hace el sistema.

El primer paso para escribir un caso de uso de forma eficiente es definir el conjunto de actores que podrán llevar las acciones a cabo. Un actor, es un elemento que se comunica con el sistema y que es externo al sistema en sí mismo, es decir la persona o sistema que utiliza el producto. En nuestro caso, debido a que el controlador Kinect será una aplicación utilizada únicamente por el profesor para controla la aplicación educativa de realidad aumentada, nuestro actor será él.

A continuación mostraremos un diagrama de casos de uso, que nos será útil para ver de forma general y muy visual cuáles serán las acciones que el actor podrá realizar en el sistema.

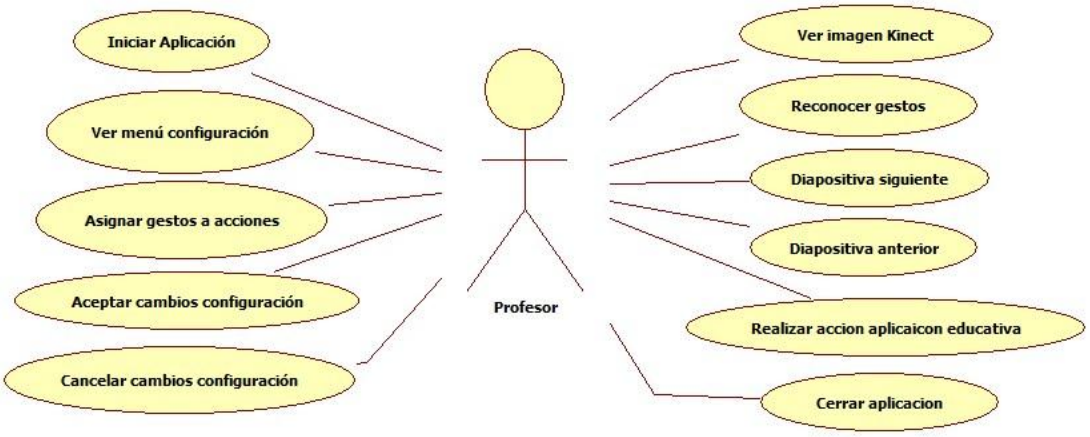


ILUSTRACIÓN 20. DIAGRAMA DE CASOS DE USO

El siguiente paso será describir de forma detallada cada uno de estos casos de uso, de manera que se pueda ver cuál es la interacción entre el actor y el sistema, qué condiciones deben darse para que el caso de uso pueda ser llevado a cabo o cuál es el flujo de interacción que da para realizar la acción. Para efectuar esta descripción de casos de uso, se tendrán en cuenta los siguientes campos:

- **Identificador.** Identificará de forma unívoca cada uno de los casos de uso. Su nomenclatura se CU-XX, donde XX será un número entero que comenzará en el valor 01 y se irá incrementando en una unidad para cada caso de uso.
- **Nombre.** Nombre que describirá de forma breve y concisa el caso de uso.
- **Autor.** Nombre del autor del caso de uso.
- **Fecha.** Fecha de creación del caso de uso.
- **Actores.** Definirá el rol del usuario que lleva a cabo la acción determinada.
- **Descripción.** Breve explicación del caso de uso.
- **Precondiciones.** Condiciones que deben darse para que el caso de uso pueda ser ejecutado de forma correcta.
- **Flujo Normal.** Define el flujo normal y exitoso de interacción entre los actores y el sistema.
- **Flujo Alternativo.** Permite definir qué hace el sistema en caso de que ocurra un caso poco frecuente o inesperado que impida que el flujo normal se lleve a cabo.
- **Postcondiciones.** Condiciones que se darán una vez se haya ejecutado el caso de uso.

En las siguientes tablas podremos ver las descripciones de todos los casos de uso mostrados en el diagrama de casos de uso mostrado anteriormente.

Identificador	**CU-01**	**Nombre**	**Iniciar Aplicación**
Autor	Estefanía Fernández	**Fecha**	28/08/2012
Actores	Profesor		
Descripción	El usuario inicia la aplicación del controlador Kinect		
Precondiciones	Disponer de la aplicación en el ordenador		
Flujo Normal	1. Ejecutar el archivo de la aplicación 2. Esperar a que cargue la aplicación		
Flujo Alternativo	No Aplica		
Postcondiciones	Se cargará la aplicación y aparecerá		

TABLA 9. CU-01. INICIAR APLICACIÓN

Identificador	CU-02	Nombre	Ver menú configuración
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permite al usuario ver el menú de configuración		
Precondiciones	La aplicación debe estar iniciada		
Flujo Normal	1. El usuario inicia la aplicación 2. El sistema muestra el menú de configuración		
Flujo Alternativo	1. El usuario pulsa el botón de configuración en la interfaz		
Postcondiciones	Se muestra el menú de configuración		

TABLA 10. CU-02. VER MENÚ CONFIGURACIÓN

Identificador	CU-03	Nombre	Asignar gestos a acciones
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permite al usuario elegir que gestos quiere utilizar para realizar cada una de las acciones existentes		
Precondiciones	Debe estar abierto el menú de configuración		
Flujo Normal	1. Abrir el menú de configuración 2. Seleccionar el gesto deseado para cada una de las acciones		
Flujo Alternativo	No Aplica		
Postcondiciones	Se habrán realizado cambios en la configuración		

TABLA 11. CU-03. ASIGNAR GESTOS A ACCIONES

Identificador	CU-04	Nombre	Aceptar cambios configuración
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permite al usuario aceptar los cambios realizados en el menú de configuración		
Precondiciones	Debe haberse realizado algún cambio en la asignación de gestos a acciones		
Flujo Normal	1. Realizar cambio en el menú de confirmación 2. Pulsar el botón aceptar 3. Se muestra ventana de información de que los cambios se han realizado de forma correcta		
Flujo Alternativo	Si no se realizan cambios, no se podrá pulsar el botón aceptar		
Postcondiciones	Se guardarán los cambios realizados		

TABLA 12. CU-04. ACEPTAR CAMBIOS CONFIGURACIÓN

Identificador	CU-05	Nombre	Cancelar cambios configuración
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permite al usuario cancelar los cambios realizados o salir del menú de configuración sin realizar cambios		
Precondiciones	Debe estar abierto el menú de configuración		
Flujo Normal	1. Pulsar el botón cancelar		
Flujo Alternativo	No Aplica		
Postcondiciones	Se descartarán los cambios realizados y se saldrá del menú de configuración		

TABLA 13. CU-05. CANCELAR CAMBIOS CONFIGURACIÓN

Identificador	CU-06	Nombre	Ver imagen Kinect
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permite al usuario ver una imagen con la imagen que captura el sensor Kinect		
Precondiciones	La aplicación debe estar iniciada		
Flujo Normal			
Flujo Alternativo	No Aplica		
Postcondiciones	El usuario podrá ver la imagen capturada por Kinect		

TABLA 14. CU-06. VER IMAGEN KINECT

Identificador	CU-07	Nombre	Reconocer Gestos
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permitirá al usuario que los gestos que realice sean reconocidos por el sensor Kinect		
Precondiciones	Deben haberse asignado gestos a las acciones		
Flujo Normal	1. El usuario realiza un gesto 2. El sistema comprueba si no hay un gesto reconocido actualmente 3. Reconoce el gesto 4. El sistema realiza la acción correspondiente		
Flujo Alternativo	2. Si existe un gesto reconocido esperará a que éste sea deshecho		
Postcondiciones	Se reconocerá el gesto realizado por el usuario		

TABLA 15. CU-07. RECONOCER GESTOS

Identificador	CU-08	Nombre	Pasar diapositiva siguiente
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permitirá al usuario pasar a la siguiente diapositiva de un PowerPoint		
Precondiciones	Debe haber reconocido el gesto asociado a esta acción		
Flujo Normal	1. El usuario realiza un gesto 2. El sistema reconoce un gesto 3. Realiza acción de pasar a la siguiente diapositiva		
Flujo Alternativo	3. Si existe un gesto anterior no realizará la acción hasta que el gesto anterior sea deshecho		
Postcondiciones	Se pasará a la siguiente diapositiva del PowerPoint		

TABLA 16. CU-08. PASAR DIAPOSITIVA SIGUIENTE

Identificador	CU-09	Nombre	Pasar diapositiva anterior
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permitirá al usuario pasar a la anterior diapositiva de un PowerPoint		
Precondiciones	Debe haber reconocido el gesto asociado a esta acción		
Flujo Normal	1. El usuario realiza un gesto 2. El sistema reconoce un gesto 3. Realiza acción de pasar a la anterior diapositiva		
Flujo Alternativo	2. Si existe un gesto anterior no realizará la acción hasta que el gesto anterior sea deshecho		
Postcondiciones	Se pasará a la anterior diapositiva del PowerPoint		

TABLA 17. CU-09. PASAR DIAPOSITIVA ANTERIOR

Identificador	CU-10	Nombre	Realizar acción de la aplicación educativa
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permitirá al usuario realizar una acción en la aplicación educativa de realidad aumentada		
Precondiciones	Haber reconocido un gesto asociado a una acción de la aplicación educativa		
Flujo Normal	1. El usuario realiza un gesto 2. El sistema reconoce el gesto y envía el dato correspondiente a la aplicación educativa a través de un socket 3. Se realiza la acción en la aplicación educativa		
Flujo Alternativo	2. Si existe un gesto anterior no realizará la acción hasta que el gesto anterior sea deshecho		
Postcondiciones	Se realizará la acción determinada en la aplicación educativa de realidad aumentada		

TABLA 18. CU-10. REALIZAR ACCIÓN EN LA APLICACIÓN EDUCATIVA

Identificador	CU-11	Nombre	Cerrar aplicación
Autor	Estefanía Fernández	Fecha	28/08/2012
Actores	Profesor		
Descripción	Permitirá al usuario cerrar la aplicación		
Precondiciones	La aplicación debe estar iniciada		
Flujo Normal	1. El usuario pulsa el botón de cerrar 2. Se cierra la aplicación		
Flujo Alternativo	No Aplica		
Postcondiciones	La aplicación se cierra		

TABLA 19. CU-11. CERRAR APLICACIÓN

5.1.3. Análisis de Requisitos

El análisis de requisitos es un punto clave en cualquier desarrollo de software. Es en este punto en el que se deben definir las necesidades del producto que se debe desarrollar. El objetivo principal, por tanto, es el de describir una serie de requisitos mediante los cuales se pueda evaluar que el producto final desarrollado se ajusta a estos requisitos, es decir, si cumple toda la funcionalidad pedida.

Los requisitos que se van a definir pueden ser divididos en dos grandes grupos: requisitos funcionales y requisitos no funcionales.

- **Requisitos Funcionales.** Son aquellos requisitos que describen el funcionamiento del sistema. De este modo, definirán el comportamiento interno del software especificando cómo los casos de uso serán llevados a cabo.
- **Requisitos No Funcionales.** Son los requisitos que especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. Dentro del ámbito de los requisitos no funcionales podemos diferenciar una gran variedad de tipos diferentes:
    - **De rendimiento.** Especifican valores numéricos para variables de rendimiento, como tasas de transferencia o velocidad de procesos.
    - **De Interfaz.** Detallan el hardware y/o software con el que el sistema debe interactuar.
    - **De operación.** Indican cómo va a realizar el sistema las tareas para las que ha sido construido, de forma que se garanticen los niveles de servicio requeridos.
    - **De recursos.** Definen los límites superiores en recursos físicos tales como potencia, espacio de disco, memoria principal, etc.
    - **De comprobación.** Definen cómo el sistema debe verificar los datos de entrada y salida.
    - **De documentación.** Describen los requisitos para la documentación del proyecto.
    - **De seguridad.** Especifican los requisitos para asegurar el sistema contra amenazas de confidencialidad, integridad y disponibilidad.
    - **De calidad.** Definen los atributos de software que aseguran que será adecuado para su propósito.
    - **De mantenimiento.** Detallan la facilidad que tendrá el software para reparar los defectos o adaptarlo a nuevos requisitos.

Además, para que el análisis de requisitos sea realizado de forma correcta, éste debe cumplir, en la medida de lo posible, una serie de características, tal y como se especifica en el estándar IEEE 830. Las características deseables para la correcta definición de los requisitos son las siguientes:

- **Corrección.** La especificación de requisitos solo será correcta si todo requisito que se refleja en ella refleja alguna necesidad real.
- **Ambigüedad.** El documento no debe ser ambiguo. Para que esto ocurra, cada uno de los requisitos descritos tiene una única interpretación y deben describirse utilizando un término único para evitar confusiones.
- **Completitud.** Los requisitos deben ser completos. Para ello, se deben incluir todos los requisitos significativos del software.

- **Verificabilidad.** Un requisito debe poder ser verificado mediante un proceso no excesivamente costoso, de modo que una persona o máquina pueda comprobar que el software satisface este requisito.
- **Consistencia.** El conjunto de requisitos no debe llevar a situaciones contradictorias o conflictivas entre ellos.
- **Clasificación.** Los requisitos deben estar clasificados según la importancia de los mismos. De este modo, se les dará una prioridad para que a la hora de implementar no se empleen excesivos recursos para requisitos menos prioritarios.
- **Modificabilidad.** La especificación de requisitos es modificable si se encuentra estructurada de forma de forma que los cambios puedan ser realizados de forma sencilla y consistente.
- **Trazabilidad.** Los requisitos serán trazables si tenemos claro en todo momento cuál es su origen y nos facilita la referencia de cada requisito a sus correspondientes componentes de diseño e implementación.

A continuación, se describirán los campos que serán utilizados para la definición de los requisitos del sistema. De este modo, sabremos qué información se incluye en cada uno de los campos.

- **Identificador.** Este campo identificará de forma unívoca cada uno de los requisitos. Este campo estará formado por un acrónimo que defina el tipo de requisito del que vamos a tratar y un número que lo identifique dentro de este grupo de requisitos. En la siguiente tabla se muestran los acrónimos utilizados para las diferentes clases de requisitos:

ACRÓNIMO	TIPO DE REQUISITO
RF	Requisito Funcional
RNFR	Requisito No Funcional de Rendimiento
RNFI	Requisito No Funcional de Interfaz
RNFO	Requisito No Funcional de Operación
RNFC	Requisito No Funcional de Comprobación
RNFD	Requisito No Funcional de Documentación
RNFM	Requisito No Funcional de Mantenimiento
RNFU	Requisito No Funcional de Usabilidad
RNFS	Requisito No Funcional de Soporte

TABLA 20. TIPOS DE REQUISITOS

- **Título.** Breve descripción del requisito.
- **Fecha.** Fecha en la que fue creado o modificado el requisito por última vez.
- **Versión.** Este campo indicará el estado del requisito en un momento dado de su desarrollo o modificación. Comenzarán a enumerarse desde el número 1, incrementando en una unidad su valor según se vayan realizando modificaciones.

- **Descripción.** Se describirá de forma concisa el objetivo del requisito. Debe ser coherente y evitar ambigüedades.
- **Prioridad.** Este campo determina la importancia de realizar el requisito respecto a los demás. Para definirlo de forma correcta se ha tenido en cuenta el siguiente rango de prioridades:
  - **Alta.** Los requisitos con esta prioridad deberán anteponerse a los demás debido a su relevancia en el proyecto.
  - **Media.** Aunque los requisitos con esta prioridad no son tan importantes como los del caso anterior, su no cumplimiento podría causar deficiencias en la calidad del producto final.
  - **Baja.** Los requisitos con prioridad baja serán los últimos en ejecutarse debido a que no son de vital importancia.
- **Estabilidad.** La estabilidad nos muestra la posibilidad de que un requisito sea modificado a lo largo del ciclo de vida del proyecto. Los niveles tenidos en cuenta son los siguientes:
  - **Alta.** El requisito no será modificado durante el proyecto.
  - **Media.** El requisito podría ser modificado durante el ciclo de vida del proyecto debido a cambios en el mismo.
  - **Baja.** El requisito puede ser modificado con asiduidad durante el desarrollo.
- **Necesidad.** Mediante este campo definiremos la utilidad del requisito, así como su influencia a lo largo del proyecto. Los niveles de necesidad son los siguientes:
  - **Esencial.** El requisito debe cumplirse obligatoriamente.
  - **Deseable.** No es obligatorio pero si recomendable, ya que aportaría calidad al proyecto.
  - **Opcional.** El requisito tiene carácter opcional, por lo que su incumplimiento no afectaría al uso normal del sistema.

5.1.3.1. Requisitos Funcionales

A continuación se describirán los requisitos funcionales mediante los datos definidos en el punto anterior.

Identificador	RF-01	Título	Iniciar aplicación
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder iniciar la aplicación mediante un ejecutable o desde el propio entorno de desarrollo		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 21. RF-01. INICIAR APLICACIÓN

Identificador	RF-02	Título	Ver menú configuración
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder ver el menú de configuración al iniciar la aplicación		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 22. RF-02. VER MENÚ CONFIGURACIÓN

Identificador	RF-03	Título	Abrir menú configuración
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder acceder al menú de configuración en cualquier momento de la ejecución del programa a través de un botón en la interfaz		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 23. RF-03. ABRIR MENÚ CONFIGURACIÓN

Identificador	RF-04	Título	Cargar configuración anterior
Versión	1	Fecha	29/08/2012
Descripción	Se deberá poder cargar la configuración guardada con anterioridad para evitar que el usuario tenga que realizar los cambios cada vez que se ejecute la aplicación		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 24. RF-04. CARGAR CONFIGURACIÓN ANTERIOR

Identificador	RF-05	Título	Asignar gesto a la acción "limpiar visión"
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder asignar un gesto a la acción "limpiar pantalla" correspondiente a la aplicación educativa de realidad aumentada. Los gestos que pueden ser seleccionados son los siguientes: • Swipe Left • Swipe Right • Left Hand Up • Right Hand Up • Left Hand Push • Right Hand Push • Hands Together		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 25. RF-05. ASIGNAR GESTO A LA ACCIÓN "LIMPIAR VISIÓN"

Identificador	RF-06	Título	Asignar gesto a la acción "activar visión"
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder asignar un gesto a la acción "activar pantalla" correspondiente a la aplicación educativa de realidad aumentada. Los gestos que pueden ser seleccionados son los siguientes: • Swipe Left • Swipe Right • Left Hand Up • Right Hand Up • Left Hand Push • Right Hand Push • Hands Together		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 26. RF-06. ASIGNAR GESTO A LA ACCIÓN "ACTIVAR VISIÓN"

Identificador	RF-07	Título	Asignar gesto a la acción "siguiente diapositiva"
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder asignar un gesto a la acción "siguiente diapositiva" correspondiente al manejo del PowerPoint. Los gestos que pueden ser seleccionados son los siguientes: • Swipe Left • Swipe Right • Left Hand Up • Right Hand Up • Left Hand Push • Right Hand Push • Hands Together		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 27. RF-07. ASIGNAR GESTO A LA ACCIÓN "SIGUIENTE DIAPOSITIVA"

Identificador	RF-08	Título	Asignar gesto a la acción "anterior diapositiva"
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder asignar un gesto a la acción "anterior diapositiva" correspondiente al manejo del PowerPoint. Los gestos que pueden ser seleccionados son los siguientes: • Swipe Left • Swipe Right • Left Hand Up • Right Hand Up • Left Hand Push • Right Hand Push • Hands Together		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 28. RF-08. ASIGNAR GESTO A LA ACCIÓN "ANTERIOR DIAPOSITIVA"

Identificador	RF-09	Título	Aceptar cambios de la configuración
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder aceptar los cambios realizados en el menú de configuración si lo considera oportuno		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 29. RF-09. ACEPTAR CAMBIOS DE LA CONFIGURACIÓN

Identificador	RF-10	Título	Cancelar cambios de la configuración
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder aceptar los cambios realizados en el menú de configuración si lo considera oportuno		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 30. RF-10. CANCELAR CAMBIOS DE LA CONFIGURACIÓN

Identificador	RF-11	Título	Guardar cambios de la configuración
Versión	1	Fecha	29/08/2012
Descripción	Se deberán poder guardar los cambios realizados en la configuración, de forma que se tenga acceso a ellos posteriormente para cargar los datos archivados		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 31. RF-11. GUARDAR CAMBIOS DE LA CONFIGURACIÓN

Identificador	RF-12	Título	Ver imagen del sensor Kinect
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder ver la imagen captada por el sensor Kinect		
Prioridad	Media		
Estabilidad	Alta		
Necesidad	Deseable		

TABLA 32. RF-12. VER IMAGEN DEL SENSOR KINECT

Identificador	RF-13	Título	Obtener posición de los Joints
Versión	1	Fecha	29/08/2012
Descripción	Se deberá poder obtener la posición de los Joints para poder calcular si se ha realizado alguno de los gestos reconocidos por el sistema		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 33. RF-13. OBTENER POSICIÓN DE LOS JOINTS

Identificador	RF-14	Título	Calcular posiciones
Versión	1	Fecha	29/08/2012
Descripción	Se deberán poder calculas las posiciones relativas de las manos en los tres ejes respecto a un punto fijo, en este caso la cabeza.		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 34. RF-14. CALCULAR POSICIONES

Identificador	RF-15	Título	Comprobar si hay gesto reconocido
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá comprobar si existe un gesto reconocido que no ha vuelto a su posición original		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 35. RF-15. COMPROBAR SI HAY GESTO RECONOCIDO

Identificador	RF-16	Título	Reconocer gesto "Swipe Left"
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá poder reconocer el gesto "Swipe Left" consistente en que la mano izquierda se encuentre a una distancia igual o superior a 0.45 metros a la izquierda de la cabeza en el eje X		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 36. RF-16. RECONOCER GESTO "SWIPE LEFT"

Identificador	RF-17	Título	Reconocer gesto "Swipe Right"
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá poder reconocer el gesto "Swipe Right" consistente en que la mano derecha se encuentre a una distancia igual o superior a 0.45 metros a la derecha de la cabeza en el eje X		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 37. RF-17. RECONOCER GESTO "SWIPE RIGHT"

Identificador	RF-18	Título	Reconocer gesto "Left Hand Up"
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá poder reconocer el gesto "Left Hand Up" consistente en que la mano izquierda se encuentre por encima de la cabeza en el eje Y		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 38. RF-18. RECONOCER GESTO "LEFT HAND UP"

Identificador	RF-19	Título	Reconocer gesto "Right Hand Up"
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá poder reconocer el gesto "Right Hand Up" consistente en que la mano derecha se encuentre por encima de la cabeza en el eje Y		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 39. RF-19. RECONOCER GESTO "RIGHT HAND UP"

Identificador	RF-20	Título	Reconocer gesto "Left Hand Push"
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá poder reconocer el gesto "Left Hand Push" consistente en que la mano izquierda se encuentre a una distancia igual o superior a 0.45 metros al frente de la cabeza en el eje Z		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 40. RF-20. RECONOCER GESTO "LEFT HAND PUSH"

Identificador	RF-21	Título	Reconocer gesto "Right Hand Push"
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá poder reconocer el gesto "Right Hand Push" consistente en que la mano derecha se encuentre a una distancia igual o superior a 0.45 metros al frente de la cabeza en el eje Z		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 41. RF-21. RECONOCER GESTO "RIGHT HAND PUSH"

Identificador	RF-22	Título	Reconocer gesto "Hands Together"
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá poder reconocer el gesto "Hands Together" consistente en que la mano izquierda y la mano derecha se junten en todos los ejes		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 42. RF-22. RECONOCER GESTO "HANDS TOGETHER"

Identificador	RF-23	Título	Cambiar estado del gesto
Versión	1	Fecha	29/08/2012
Descripción	Se deberá poder cambiar el estado del gesto de activo a inactivo de forma que solo pueda existir un gesto activo a la vez		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 43. RF-23. CAMBIAR ESTADO DDEL GESTO

Identificador	RF-24	Título	Pasar a diapositiva siguiente
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder pasar a la diapositiva siguiente de un PowerPoint abierto mediante el gesto asignado para tal fin en el menú de configuración		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 44. RF-24. PASAR A DIAPOSITIVA SIGUIENTE

Identificador	RF-25	Título	Pasar a diapositiva anterior
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder pasar a la diapositiva anterior de un PowerPoint abierto mediante el gesto asignado para tal fin en el menú de configuración		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 45. RF-25. PASAR A DIAPOSITIVA ANTERIOR

Identificador	RF-26	Título	Enviar dato por socket
Versión	1	Fecha	29/08/2012
Descripción	Se deberá poder enviar datos por un socket a la aplicación educativa de realidad aumentada al reconocer un gesto asignado a acciones relativas a ella		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 46. RF-26. ENVIAR DATO POR SOCKET

Identificador	RF-27	Título	Cerrar aplicación
Versión	1	Fecha	29/08/2012
Descripción	El usuario deberá poder cerrar la aplicación		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 47. RF-27. CERRAR APLICACIÓN

5.1.3.2. Requisitos No Funcionales

A continuación se describirán los requisitos no funcionales contemplados para la realización del proyecto. Esta definición se realizará de forma similar al caso de los requisitos funcionales, utilizando los mismos campos para su especificación.

Identificador	RNFR-01	Título	Tiempo de respuesta
Versión	1	Fecha	29/08/2012
Descripción	La aplicación deberá ejecutarse en tiempo real con un retardo máximo de 2 milisegundos de forma que no influya en la ejecución de la misma		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 48. RNFR-01. TIEMPO DE RESPUESTA

Identificador	RNFI-01	Título	Compatibilidad con programas externos
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá ser compatible con programas externos. Deberá ser compatible con PowerPoint y la aplicación educativa que queremos controlar, con las que interaccionará mediante los gestos reconocidos por nuestra aplicación		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 49. RNFI-02. COMPATIBILIDAD CON PROGRAMAS EXTERNOS

Identificador	RNFI-02	Título	Interfaz de conexión XML
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá ser capaz de leer y escribir en un fichero XML en el que se guardarán las asignaciones de los gestos con las acciones		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 50. RNFI-02. INTERFAZ DE CONEXIÓN XML

Identificador	RNFI-03	Título	Interfaz gráfica
Versión	1	Fecha	29/08/2012
Descripción	El sistema dispondrá de una interfaz gráfica sencilla e intuitiva, con colores amigables. Para ello se utilizará una ventana similar a las del sistema operativo Windows, de modo que sea familiar al usuario		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 51. RNFI-03. INTERFAZ GRÁFICA

Identificador	RNFI-04	Título	Contenido
Versión	1	Fecha	29/08/2012
Descripción	El contenido del menú de la aplicación debe tener las opciones claras y ser conciso, de forma que no pueda llevar a confusión al usuario		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 52. RNFI-04. CONTENIDO

Identificador	RNFO-01	Título	Acceso a la aplicación
Versión	1	Fecha	29/08/2012
Descripción	El acceso a la aplicación podrá ser llevado a cabo a través de un fichero ejecutable o iniciando el proceso desde el propio entorno de desarrollo		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 53. RNFO-01. ACCESO A LA APLICACIÓN

Identificador	RNFC-01	Título	Comprobación de asignación de gestos
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá comprobar que no se permita asignar un mismo gesto a diferentes acciones		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 54. RNFC-01. COMPROBACIÓN DE ASIGNACIÓN DE GESTOS

Identificador	RNFC-02	Título	Comprobación de gesto activo
Versión	1	Fecha	29/08/2012
Descripción	Se deberá comprobar si existe algún gesto activo a la hora de reconocer uno nuevo, ya que solo puede ser reconocido uno a la vez para evitar problemas y confusiones		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 55. RNFC-02. COMPROBACIÓN DE GESTO ACTIVO

Identificador	RNFD-01	Título	Idioma de la documentación
Versión	1	Fecha	29/08/2012
Descripción	Debido a que su uso será meramente académico dentro del ámbito de una universidad española, el idioma de la documentación será íntegramente en castellano		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 56. RNFD-01. IDIOMA DE LA DOCUMENTACIÓN

Identificador	RNFM-01	Título	Diseño modular
Versión	1	Fecha	29/08/2012
Descripción	El sistema seguirá un diseño modular, de forma que resulte más sencillo añadir funcionalidad o corregir errores que puedan darse		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 57. RNFM-01. DISEÑO MODULAR

Identificador	RNFU-01	Título	Facilidad de uso
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá ser fácil de usar por un usuario que no posea conocimientos avanzados de informática		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 58. RNFU-01. FACILIDAD DE USO

Identificador	RNFU-02	Título	Facilidad de aprendizaje
Versión	1	Fecha	29/08/2012
Descripción	El sistema deberá ser fácil de aprender, es decir, que el usuario sepa cómo utilizarlo en un plazo máximo de unas horas		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 59. RNFU-02. FACILIDAD DE APRENDIZAJE

Identificador	RNFS-01	Título	Plataforma
Versión	1	Fecha	29/08/2012
Descripción	La aplicación funcionará únicamente plataformas con sistema operativo Windows		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 60. RNFS-01. PLATAFORMA

Identificador	RNFS-02	Título	Versión
Versión	1	Fecha	29/08/2012
Descripción	La aplicación únicamente funcionará en la versión Windows 7		
Prioridad	Alta		
Estabilidad	Alta		
Necesidad	Esencial		

TABLA 61. RNFS-02. VERSIÓN

5.1.4. Diagrama de Actividad

Mediante el siguiente diagrama se expondrá de forma muy visual el funcionamiento general de la aplicación, desde el punto inicial hasta el punto final. En él se podrán ver las diferentes decisiones u opciones posibles durante la ejecución de la aplicación.

Al arrancar la aplicación, se cargan los datos de la configuración anterior y se muestran en el propio menú de configuración. En este punto el usuario tiene dos opciones, si no realiza ninguna modificación podrá pulsar el botón cancelar y la configuración quedará con la que se había cargado anteriormente. Por otro lado, el usuario puede decir realizar cambios en la asignación de gestos a las acciones, de modo que al aceptar, los nuevos datos se guarden en un fichero XML.

Una vez se ha realizado cualquiera de las dos opciones en el menú de configuración, se mostrará una pantalla con la imagen capturada por el sensor Kinect. Desde aquí comenzaría el proceso de reconocimiento de gestos. Mediante los datos obtenidos por el sensor Kinect, se comprueba en tiempo real la posición de los diferentes Joints del cuerpo. A continuación, se calcula la posición relativa de las manos respecto a la cabeza y se comprueba si se cumple alguna de las opciones de los gestos que se reconocen. Si no existe una coincidencia se volverá al punto de comprobación de la posición de los Joints. En el caso de que si se encuentren coincidencias, primero deberemos comprobar si existe algún gesto activo, ya que no pueden darse dos gestos a la vez. Si ya existe un gesto activo, volveremos al punto de comprobar la posición de los Joints. Si no existe un gesto activo, pasaremos a reconocer el gesto y realizar la acción oportuna.

En cuanto a las acciones, podemos diferenciar dos tipos. Por un lado, las acciones propias de la aplicación educativa, para las cuales se enviará un dato a través de un socket. Por otro lado, las acciones de control de las diapositivas de un PowerPoint, que realizarán su acción determinada según el gesto reconocido.

Destacar que en cualquier momento de la ejecución del programa se podrá acceder al menú de configuración y se podrá abandonar la aplicación.

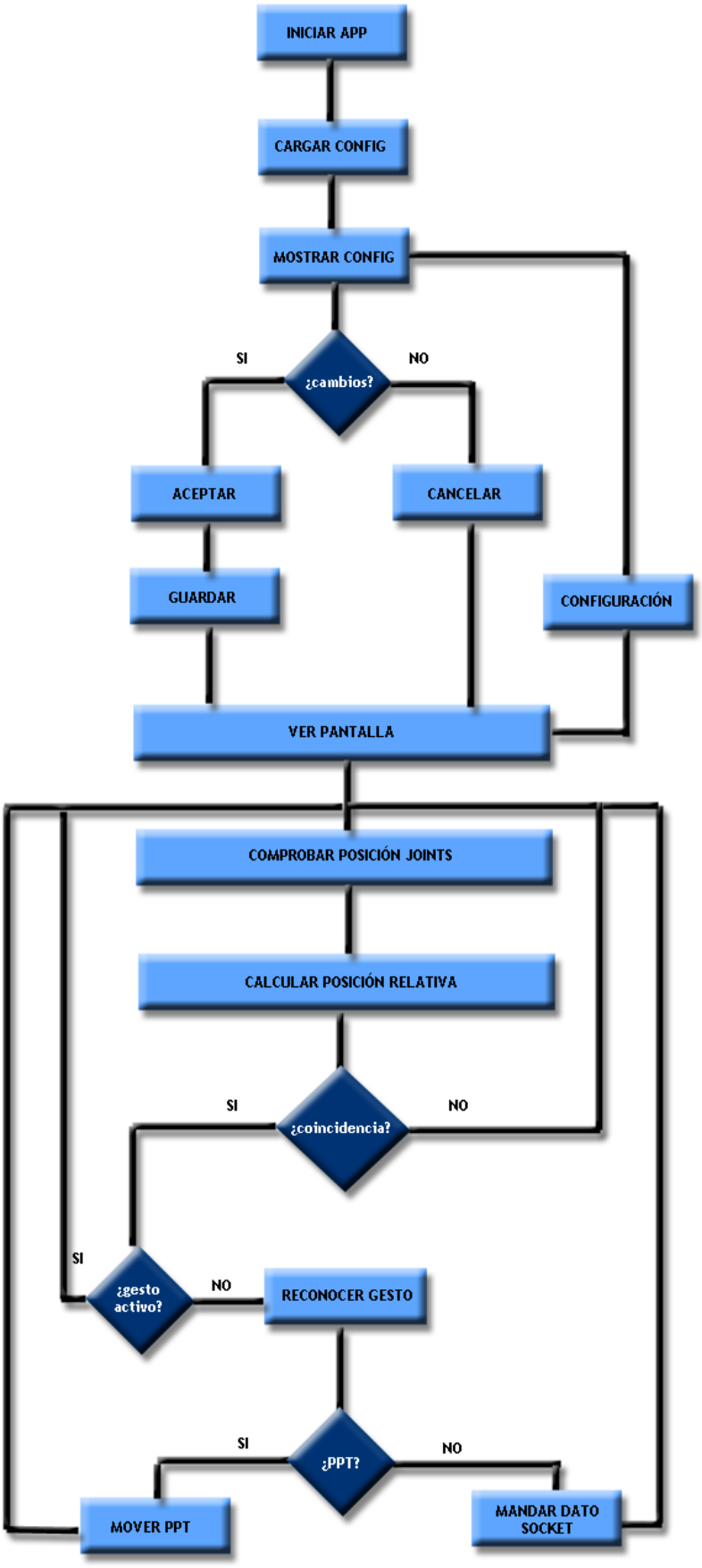


ILUSTRACIÓN 21. DIAGRAMA DE ACTIVIDAD

5.2. DISEÑO

El objetivo de este capítulo será el de solucionar los problemas planteados en el análisis anterior. Para ello, se describirá de forma pormenorizada la arquitectura del sistema y sus diferentes módulos. El diseño se realizará para todos los componentes de la aplicación, de forma que ésta quede bien definida. Además, definiremos los prototipos de las interfaces que serán utilizadas en la ejecución del proyecto.

5.2.1. Arquitectura

A continuación se mostrará la arquitectura que se utiliza para la realización del proyecto. De este modo, podremos ver de una forma más precisa la estructura que sigue la aplicación y que facilitará el proceso de implementación.

En la siguiente imagen, se muestra la arquitectura general de la aplicación, de modo que se pueden observar las interacciones realizadas entre la aplicación para Kinect realizada y los elementos externos que forman parte del sistema general. Así, vemos como el usuario que utiliza el programa, se comunica con nuestra aplicación Kinect mediante gestos, teniendo en este punto dos objetivos diferenciados. Por un lado, encontramos el control de una presentación Power Point y, por otro, el control de una aplicación educativa de realidad aumentada.

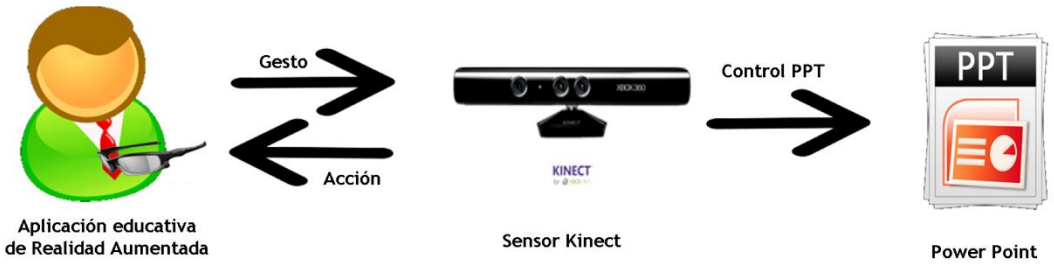


ILUSTRACIÓN 22. ARQUITECTURA GENERAL DEL SISTEMA

El siguiente paso, será describir de forma más detallada la estructura interna de nuestra aplicación en sí, de modo que se puedan conocer los diferentes módulos de los que se compone la aplicación para el sensor Kinect. A continuación se explicará de forma más específica la función y componentes de los que dispone cada uno de estos módulos. Para cada uno de ellos se describirá su propósito y las funciones que debe cumplir para que su funcionamiento sea completo y correcto.

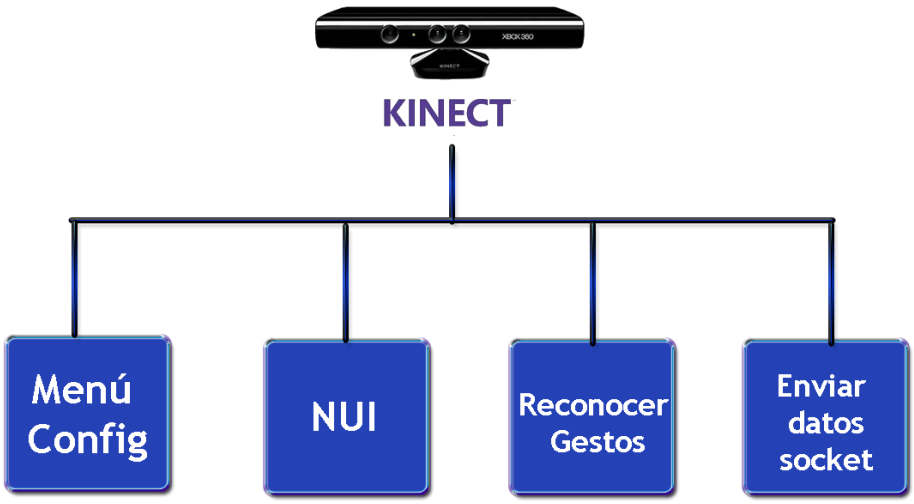


ILUSTRACIÓN 23. ARQUITECTURA DE LA APLICACIÓN KINECT

Módulo	Menú Configuración
Propósito	Su objetivo es el de proporcionar al usuario una interfaz gráfica en la que pueda definir qué gestos quiere utilizar para las opciones disponibles. De este modo, podrá realizar el control de la aplicación externa de forma totalmente personalizada
Funciones	• Deberá proporcionar una interfaz de usuario sencilla y fácil de aprender • Deberá cargar la configuración guardada con anterioridad si existiera • Deberá permitir a los usuarios modificar los gestos asignados a cada una de las acciones • Deberá permitir al usuario aceptar los cambios realizados • Deberá permitir al usuario cancelar los cambios realizados • Deberá guardar los cambios que se hayan realizado en la configuración

TABLA 62. MÓDULO 1. MENÚ CONFIGURACIÓN

Módulo	NUI
Propósito	El objetivo de este módulo es el de inicializar la Interfaz Natural de Usuario o NUI, de modo que permita que la aplicación obtener la imagen obtenida por el sensor Kinect para poder capturar los movimientos realizados por el usuario
Funciones	• Deberá iniciar la captura de imagen del sensor • Deberá indicar al usuario si no se dispone de un sensor Kinect reconocido por el computador • Deberá activar los sensores del dispositivo • Deberá definir qué tipo de imagen queremos visualizar • Deberá definir la resolución de imagen que nos proporcionará el sensor Kinect • Deberá activar el reconocimiento del esqueleto • Deberá definir el ángulo de elevación del sensor Kinect • Deberá desactivar el sensor Kinect en caso de que se cierre la aplicación

TABLA 63. MÓDULO 2. NUI

Módulo	Reconocedor de Gestos
Propósito	Su objetivo es el de reconocer los gestos que realice el usuario a través del sensor Kinect, de modo que puedan realizarse las acciones relacionadas con estos
Funciones	• Deberá activar el rastreo de los *Joints* de la cabeza, mano izquierda y mano derecha, que serán los utilizados para los gestos que se van reconocer • Deberá obtener las posiciones de los *Joints* • Deberá calcular las posiciones relativas de los *Joints* hasta encontrar una coincidencia con los gestos que se contemplan • Deberá comprobar si existen un gesto activo actualmente • Si existe un gesto activo, volverá a obtener las posiciones de los Joints • Si no existe un gesto activo, tomará el gesto como reconocido y cambiará su estado a activo • Cuando el gesto es reconocido, realizará la acción oportuna

TABLA 64. MÓDULO 3. RECONOCEDOR DE GESTOS

Módulo	Enviar datos por Socket
Propósito	Su objetivo es el de crear un socket que proporcione un método de comunicación entre la aplicación de Kinect y la aplicación educativa de realidad aumentada
Funciones	• Crear un socket • Deberá enviar el dato correspondiente al gesto reconocido a través de un socket • Cerrar socket

TABLA 65. MÓDULO 4. ENVIAR DATO POR SOCKET

5.2.2. Interfaces

En este punto se tratará de detallar mediante un prototipo cómo serán las interfaces gráficas de la aplicación. En este caso, solo se dispondrá de dos interfaces. Por un lado, la interfaz que nos presente el menú de configuración y nos permita la asignación de gestos a las acciones. Por otro lado, disponemos de una interfaz que nos permita observar la imagen capturada por el sensor Kinect y que nos permita acceder al menú de configuración en cualquier momento.

5.2.2.1. Menú de Configuración

El menú de configuración dispondrá de distintos elementos. Por una parte, tendrá un campo de texto en el que se describa brevemente la finalidad de la interfaz, es decir, que explique al usuario brevemente que puede modificar la configuración para que se adapte a sus necesidades.

La interfaz dispone además de diferentes campos que incluirán las acciones que es posible realizar mediante el uso de la aplicación. A su lado se tendrá una lista desplegable con los diferentes gestos que pueden ser asociados a las acciones.

Para finalizar, se tendrán dos botones que permitirán al usuario aceptar o cancelar los cambios realizados en la configuración.

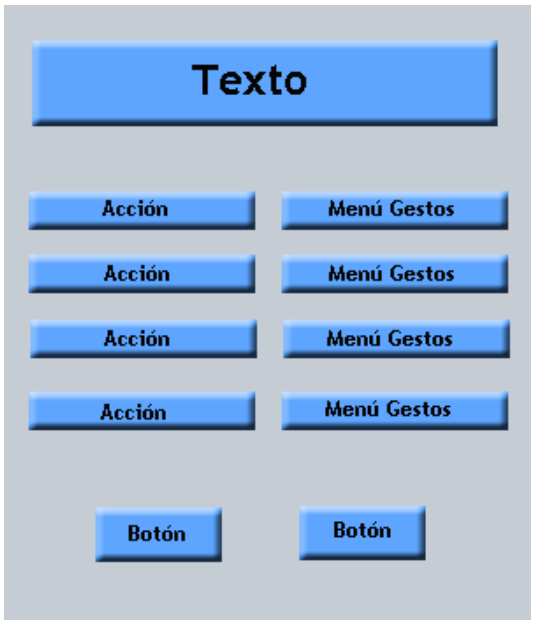


ILUSTRACIÓN 24. PROTOTIPO DE INTERFAZ DEL MENÚ DE CONFIGURACIÓN

5.2.2.2. Pantalla Inicial

La pantalla inicial es en la que se mostrará la imagen captada por el sensor Kinect. Su estructura es muy sencilla, se compone de la propia imagen obtenida y un botón que nos permita acceder al menú de configuración en cualquier momento de la ejecución de la aplicación.

ILUSTRACIÓN 25. PROTOTIPO DE LA PANTALLA INICIAL

5.3. IMPLEMENTACIÓN

En esta sección se explicarán detalladamente los aspectos más relevantes de la implementación de la aplicación para Kinect. Haciendo uso de los puntos anteriores en los que se especifican los diferentes requisitos que se deben cumplir, así como el diseño que debe poseer el sistema, se implementarán las diferentes funcionalidades que harán que nuestra aplicación funcione de la manera que se espera.

La implementación se basa en dos clases fundamentales. Por un lado, la que nos permite crear y dar funcionalidad al menú de configuración. En este caso, como veremos a continuación, la creación de la propia interfaz se realiza de forma visual en el entorno de desarrollo, y es este entorno el que nos genera el código de la interfaz. El trabajo de implementación de este menú de configuración se encontrará en añadir funcionalidad a los diferentes botones y campos de texto que se utilizan en la interfaz. Por otro lado, se dispone de la clase principal, que será la encargada de dar el resto de la funcionalidad a la aplicación, de modo que será el núcleo de la implementación.

5.3.1. Menú de Configuración

Como se ha comentado, la creación de la interfaz gráfica del menú de configuración se realiza de forma visual a través del entorno de desarrollo, generando un código automáticamente con las características de los elementos incluidos en la interfaz. Este código nos permitirá inicializar todos los componentes de la interfaz gráfica mediante un método *InitializeComponent ().* Este método incluye todos los datos de localización, nombre, tamaño, texto que incluyen o eventos que pueden producirse en ellos como veremos a continuación.

El primer paso es crear todos los elementos que queremos incorporar a la interfaz, como se muestra a continuación.

```
this.AcceptButton = new System.Windows.Forms.Button();
this.CancelButton = new System.Windows.Forms.Button();
this.label1 = new System.Windows.Forms.Label();
this.Action1 = new System.Windows.Forms.Label();
this.Action2 = new System.Windows.Forms.Label();
this.Action3 = new System.Windows.Forms.Label();
this.Action4 = new System.Windows.Forms.Label();
this.Gesture1 = new System.Windows.Forms.ComboBox();
this.Gesture2 = new System.Windows.Forms.ComboBox();
this.Gesture3 = new System.Windows.Forms.ComboBox();
this.Gesture4 = new System.Windows.Forms.ComboBox();
this.SuspendLayout();
```

CÓDIGO 1. CREACIÓN DE LOS ELEMENTOS DE LA INTERFAZ

A continuación se especifican las características de cada uno de los elementos de la interfaz. Para evitar ser repetitivos se incluirá solo un caso a modo de ejemplo, ya que para el resto de los objetos serán los mismos casos.

```
//
// Action1
//
this.Action1.AutoSize = true;
this.Action1.Location = new System.Drawing.Point(41, 63);
this.Action1.Name = "Action1";
this.Action1.Size = new System.Drawing.Size(49, 13);
this.Action1.TabIndex = 3;
this.Action1.Text = "Acción 1";
```

CÓDIGO 2. EJEMPLO DE ESPECIFICACIÓN DE LAS CARACTERÍSTICAS DE LOS ELEMENTOS DE LA INTERFAZ

En cuanto a la funcionalidad de los diferentes elementos, cabe destacar el uso de ComboBox, que son campos desplegables que nos muestran los diferentes gestos que se pueden seleccionar para asignar a una acción concreta. Se ha creado un array con los diferentes gestos existentes para rellenar las opciones de los ComboBox. A continuación, una vez se han iniciado estos elementos con un valor determinado, éstos deben borrarse del resto de componentes, para asegurarnos así que no pueden darse valores repetidos a acciones diferentes. Para finalizar, hay que tener en cuenta que pueden realizarse modificaciones en los gestos, de modo que se han creado una serie de funciones que nos permitan modificar los campos incluidos en los ComboBox, de modo que si se realiza un cambio, éste quede reflejado en todos los campos desplegables.

```
private void Gesture1_SelectedIndexChanged(object sender, EventArgs e)
{
    string gestoActual1 = (string)Gesture1.SelectedItem;


    Gesture2.Items.Remove(gestoActual1);
    Gesture2.Items.Add(gesto1);
    Gesture3.Items.Remove(gestoActual1);
    Gesture3.Items.Add(gesto1);
    Gesture4.Items.Remove(gestoActual1);
    Gesture4.Items.Add(gesto1);
    gesto1 = gestoActual1;
}
```

CÓDIGO 3. EJEMPLO DE EVENTO DE CAMBIO DE OPCIÓN EN EL COMBOBOX

Para finalizar, hay que darles la funcionalidad oportuna a los botones de aceptar y cancelar. El caso del segundo es muy simple, ya que únicamente debe ocultar la ventana. En el caso del botón aceptar, se mostrará un mensaje informativo de que los cambios se han realizado de forma correcta, de manera que se tenga informado al usuario en todo momento.

```
private void AcceptButton_Click(object sender, EventArgs e)
{

    EscribirDatos();
    MessageBox.Show("Los cambios se han realizado correctamente", "Configuración", MessageBoxButtons.OK, MessageBoxIcon.Information);
    this.Hide();
}

private void CancelButton_Click(object sender, EventArgs e)
{
    this.Hide();
}
```

CÓDIGO 4. ACCIONES DE LOS BOTONES ACEPTAR Y CANCELAR

5.3.2. Programa Principal

Es en esta clase en la que se trata el núcleo de la funcionalidad de la aplicación. Así, veremos como inicializar la NUI para poder ver la imagen captada por el sensor Kinect o cómo se reconocen los gestos realizados por el usuario.

En esta clase, se tendrán una serie de variables que nos indicarán si existe algún gesto activo, de modo que solo pueda ser ejecutado uno cada vez. Estas variables serán inicializadas a *false*, de modo que al inicio de la aplicación no existe ningún gesto activo, tal y como se ve a continuación.

```
bool isForwardGestureActive = false;
bool isBackGestureActive = false;
bool isRightHandUpActive = false;
bool isLeftHandUpActive = false;
bool isRightHandPushActive = false;
bool isLeftHandPushActive = false;
bool areHandsTogether = false;
```

CÓDIGO 5. INICIALIZACIÓN DE VARIABLES DE GESTOS ACTIVOS

A continuación, lanzamos el menú de configuración, ya que debe aparecer al inicio de la aplicación para que el usuario sepa o modifique los gestos a utilizar durante la ejecución.

```
Settings conf = new Settings();
conf.ShowDialog();
```

CÓDIGO 6. LANZAMIENTO DEL MENÚ DE CONFIGURACIÓN

El siguiente paso es cargar el evento encargado de iniciar la ventana principal en la que se podrá ver la imagen capturada por el sensor Kinect.

```
this.Loaded += new RoutedEventHandler(MainWindow_Loaded);
```

CÓDIGO 7. EVENTO QUE CARGA LA PANTALLA PRINCIPAL

Dentro de la función *MainWindow_Loaded*, se inicializa el sensor y se comprueba si existe un sensor Kinect activo en el ordenador. A continuación, se inicia el sensor Kinect mediante la función *Start ().*

```
sensor = KinectSensor.KinectSensors.FirstOrDefault();

if (sensor == null)
{
    MessageBox.Show("This application requires a Kinect sensor.");
    this.Close();
}

sensor.Start();
```

CÓDIGO 8. INICIALIZACIÓN DEL SENSOR KINECT

Además, en esta misma función, se habilitan las cámaras RGB y de profundidad definiendo la resolución a utilizar, así como el tracking del esqueleto mediante la función *Enable*. También en este punto se lanzan los eventos *sensor_ColorFrameReady* y *sensor_SkeletonFrameReady* que definirán el resto de opciones de las cámaras y los Joints que serán rastreados respectivamente.

```
sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
sensor.ColorFrameReady += new EventHandler<ColorImageFrameReadyEventArgs>(sensor_ColorFrameReady);

sensor.DepthStream.Enable(DepthImageFormat.Resolution320x240Fps30);
sensor.SkeletonStream.Enable();
sensor.SkeletonFrameReady += new EventHandler<SkeletonFrameReadyEventArgs>(sensor_SkeletonFrameReady);
```

CÓDIGO 9. HABILITAR CÁMARAS Y LANZAMIENTO DE EVENTOS

Para finalizar esta función, se lanzará un evento de cierre de la aplicación que se ejecutará cuando ésta sea cerrada.

```
Application.Current.Exit += new ExitEventHandler(Current_Exit);
```

CÓDIGO 10. LANZAMIENTO DEL EVENTO DE CIERRE DE LA APLICACIÓN

Este último evento, desactiva el sensor Kinect al cerrar la aplicación del siguiente modo.

```
void Current_Exit(object sender, ExitEventArgs e)
{
    if (sensor != null)
    {

        sensor.Stop();
        sensor.Dispose();
        sensor = null;
    }
}
```

CÓDIGO 11. EVENTO DE CIERRE DE LA APLICACIÓN

El evento *sensor_SkeletonFrameReady* resulta interesante, ya que establece los Joints que serán utilizados, en este caso la cabeza, la mano izquierda y la mano derecha y hace la llamada a la función encargada de procesar o reconocer los gestos.

```
var head = closestSkeleton.Joints[JointType.Head];
var rightHand = closestSkeleton.Joints[JointType.HandRight];
var leftHand = closestSkeleton.Joints[JointType.HandLeft];
```

CÓDIGO 12. JOINTS UTILIZADOS EN LA APLICACIÓN

```
ProcessGesture (head, rightHand, leftHand);
```

CÓDIGO 13. LLAMADA A LA FUNCIÓN ENCARGADA DEL RECONOCIMIENTO DE GESTOS

6. PRUEBAS

En este punto se incluirán diferentes pruebas para comprobar que la aplicación funciona de la forma correcta. Se dividirán las pruebas en dos grandes bloques. Por un lado, se realizarán las pruebas relacionadas con el menú de configuración. Por otro lado, se detallarán las pruebas del reconocimiento de gestos.

6.1. PRUEBAS DEL MENÚ DE CONFIGURACIÓN

Dado que no se puede asignar el mismo gesto a diferentes acciones, en los ComboBox que aparecen en la interfaz únicamente deben aparecer los aquellas opciones que puedan ser seleccionadas, de modo que evitemos dicho problema. En la siguiente prueba, comprobaremos que esto se realiza de forma correcta. Como se puede observar en la imagen, el desplegable solo indica las opciones que puede seleccionarse, de modo que podemos decir que la funcionalidad es correcta.

ILUSTRACIÓN 26. PRUEBA DATOS COMBOBOX

Si se produce algún cambio en la configuración, se deben actualizar todas las opciones de los ComboBox de forma que nunca pueda seleccionarse el mismo gesto para dos acciones diferentes. Bajo la imagen anterior, cambiaremos el gesto asignado a la acción “Anterior PPT” a “Right Hand Up” y comprobaremos de nuevo en el último ComboBox como los datos de la lista han sido actualizados.

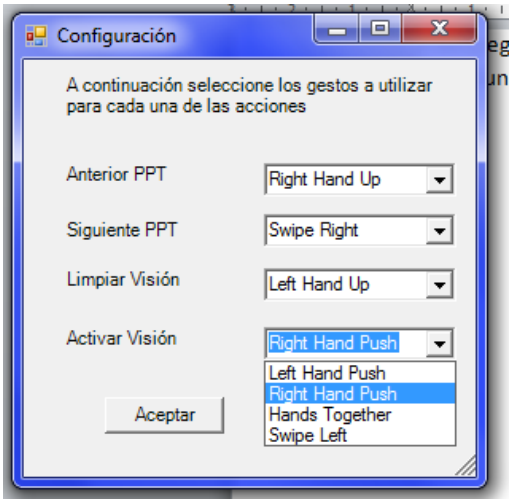


ILUSTRACIÓN 27. PRUEBA MODIFICACIÓN DEL COMBOBOX

Otra de las funcionalidades que tiene el menú de configuración, es la de cargar la configuración anterior, para evitar que el usuario debe modificar las opciones cada vez que inicie la aplicación. En las siguientes imágenes veremos los datos que contiene el fichero que almacena la configuración (se trata de un fichero de texto que contiene el nombre de los gestos asignados en el mismo orden en el que aparecen las acciones en el menú) y la ejecución del menú. Como se puede observar, los datos del fichero se corresponden con los datos que muestra la interfaz.

```
1  Swipe Left
2  Swipe Right
3  Left Hand Up
4  Right Hand Push
5
```

ILUSTRACIÓN 28. DATOS DEL FICHERO

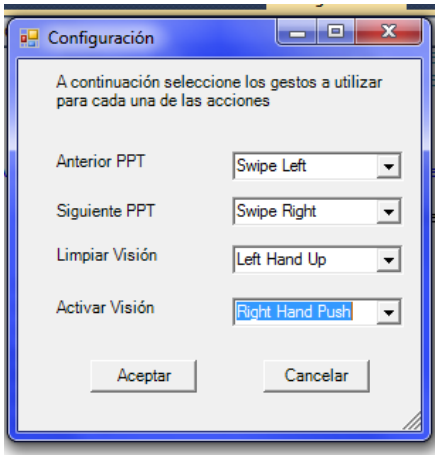


ILUSTRACIÓN 29. DATOS MOSTRADOS EN LA INTERFAZ

Para finalizar, el menú de configuración debe ser capaz de guardar las modificaciones realizadas en la interfaz. Así, sobre los datos de la prueba anterior, realizaremos varias modificaciones y después comprobaremos los resultados en el fichero de texto. En las siguientes imágenes podemos ver como los cambios se realizan correctamente.

ILUSTRACIÓN 30. MENÚ DE CONFIGURACIÓN TRAS LOS CAMBIOS

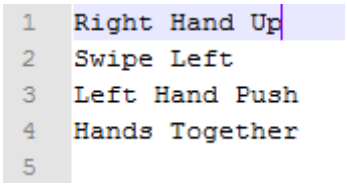


ILUSTRACIÓN 31. FICHERO DE TEXTO TRAS LA MODIFICACIÓN

6.2. PRUEBAS DE RECONOCIMIENTO DE GESTOS

Para poder comprobar el correcto funcionamiento del reconocimiento de gestos, se ha añadido al código un mensaje que aparecerá por pantalla al reconocerse el gesto que incluirá el gesto que ha sido reconocido y la acción a la que está asociado para comprobar su correcto funcionamiento. Dado que únicamente es posible ejecutar los gestos que han sido asignados a alguna acción y con el fin de probar todos los gestos, realizaremos las pruebas en dos bloques. En primer lugar, utilizaremos la última configuración guardada en el programa, la vista en la Ilustración 30, reconociendo así los gestos que en ella se muestran. A continuación, se realizarán las modificaciones oportunas en la configuración para que podamos probar el resto de los gestos.

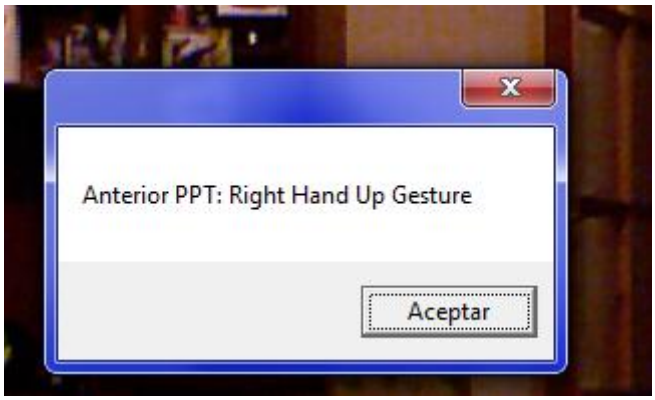


ILUSTRACIÓN 32. ACCIÓN "ANTERIOR PPT" – GESTO "RIGHT HAND UP"

ILUSTRACIÓN 33. ACCIÓN "SIGUIENTE PPT" – GESTO "SWIPE LEFT"

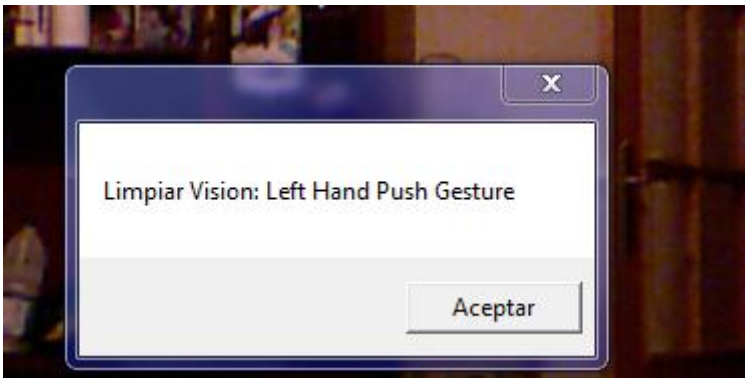


ILUSTRACIÓN 34. ACCIÓN "LIMPIAR VISIÓN" – GESTO "LEFT HAND PUSH"

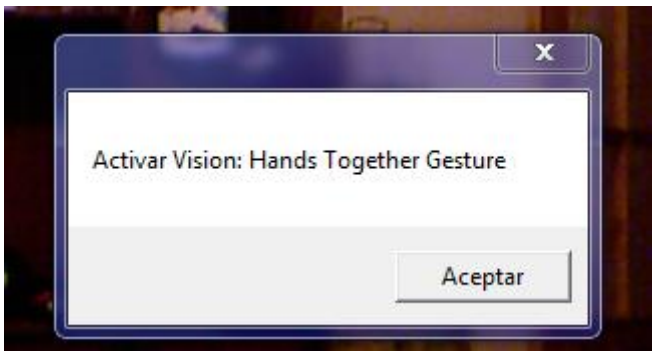


ILUSTRACIÓN 35. ACCIÓN "ACTIVAR VISIÓN" – GESTO "HANDS TOGETHER"

Como se ha comentado anteriormente, ahora se cambiarán las opciones de configuración, de modo que podamos probar los tres gestos restantes. La nueva configuración es la siguiente:

ILUSTRACIÓN 36. NUEVA CONFIGURACIÓN PARA LA PRUEBA DE RECONOCIMIENTO DE GESTOS

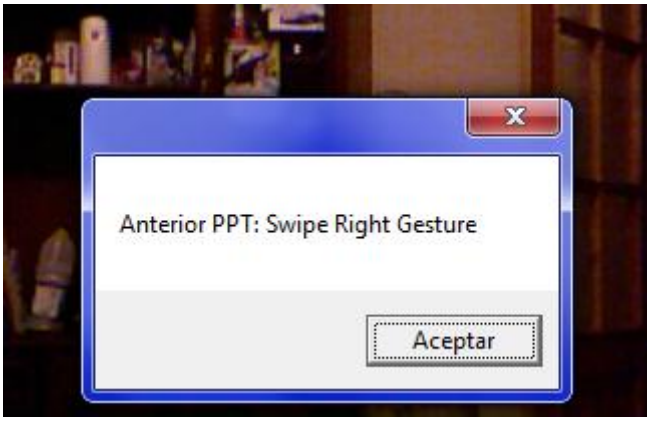


ILUSTRACIÓN 37. ACCIÓN "ANTERIOR PPT" – GESTO "SWIPE RIGHT"

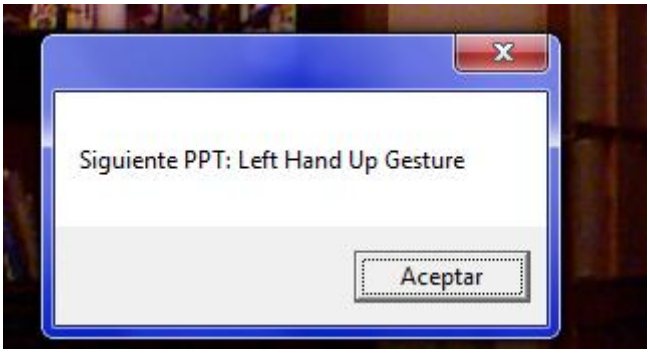


ILUSTRACIÓN 38. ACCIÓN "SIGUEINTE PPT" – GESTO "LEFT HAND UP"

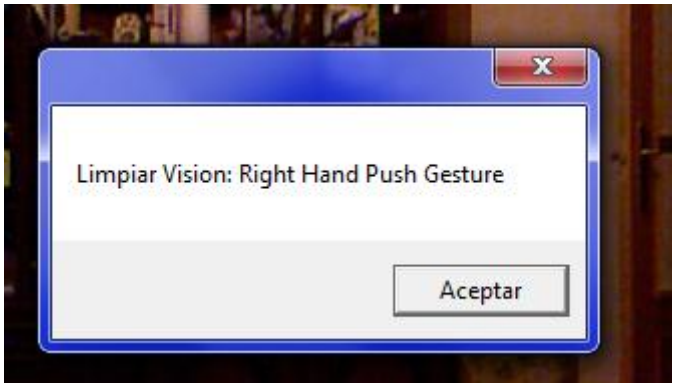


ILUSTRACIÓN 39. ACCIÓN "LIMPIAR VISIÓN" – GESTO "RIGHT HAND PUSH"

Finalmente, con la siguiente imagen comprobaremos que se puede acceder en todo momento al menú de configuración mediante un botón situado en la esquina superior derecha de la imagen capturada por el sensor Kinect.

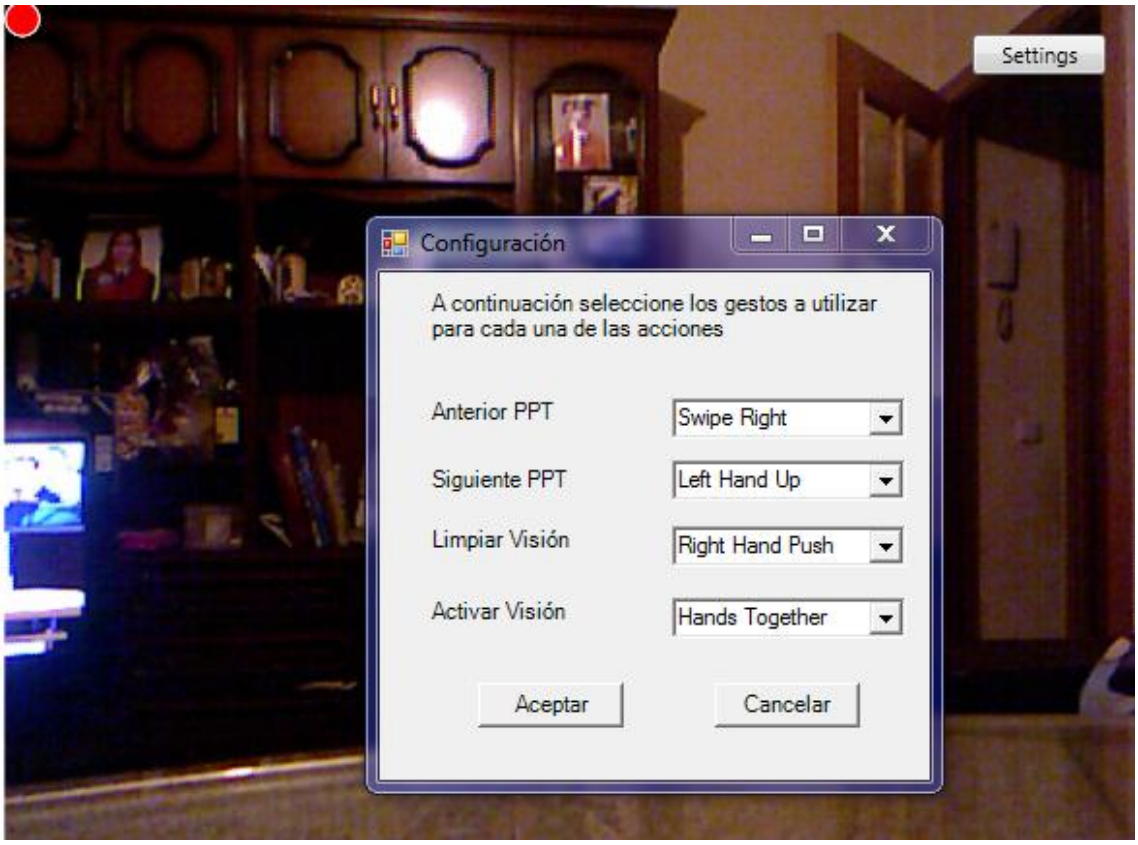


ILUSTRACIÓN 40. MENÚ CONFIGURACIÓN A TRAVÉS DEL BOTÓN EN LA PANTALLA

7. CONCLUSIONES

Una vez finalizado el desarrollo de la aplicación, se ha evaluado si la aplicación cumple los objetivos propuestos al inicio de este documento. Como veremos a continuación, las conclusiones son satisfactorias, debido a que los objetivos han sido logrados en su totalidad.

7.1. RECONOCIMIENTO DE GESTOS

El objetivo principal de este proyecto es el de crear una aplicación capaz de reconocer gestos simples del cuerpo para controlar una aplicación externa. La aplicación conseguida es capaz de reconocer un total de siete gestos sencillos, que pueden ser seleccionados por los usuarios para realizar las cuatro acciones tenidas en cuenta, entre las que se encuentra realizar el control de una presentación de diapositivas en Power Point.

7.2. CONOCER LOS DIFERENTES DISPOSITIVOS EXISTENTES

Al margen del objetivo principal, existen una serie de objetivos secundarios que han resultado muy útiles a la hora de llevar a cabo el proyecto. Uno de estos objetivos ha sido el de tener un conocimiento de otros dispositivos similares a Kinect, que nos permitieran también llevar a cabo acciones similares.

En este punto, no solo hemos aprendido el funcionamiento básico de dispositivos actuales como el WiiMote de la videoconsola Wii o el PlayStation Move de PS3, sino que también hemos conocido algunos dispositivos que son considerados antecesores de éstos y que fueron poco populares en su época debido principalmente a las tecnologías existentes. Así, se ha visto el Eye Toy de PS2, que aunque tuvo un éxito mayor que los otros dos, resultaba restrictivo en muchos casos, el Sega Activator que nos permitía manejar los juegos mediante gestos del cuerpo, aunque realmente lo que hacía era una correspondencia entre los haces de luz y los botones de un pad normal, y por último, el VRU que nos proporcionaba reconocimiento de voz muy restrictivo y al que no se le sacó provecho más allá de un par de juegos.

7.3. CONOCER EL SENSOR KINECT

Otro de los objetivos de este proyecto era el de conocer en profundidad qué es y cómo funciona el sensor Kinect de modo que nos fuera más sencillo entender el desarrollo de aplicaciones para este dispositivo.

Así pues, se han aprendido las partes de las que dispone el sensor Kinect y cómo funcionan sus diferentes cámaras y micrófonos para definir una imagen tridimensional de la escena captada por el sensor, ofreciéndonos las distancias de los objetos y permitiéndonos reconocer cuando una persona se encuentra en el campo de visión del sensor para que pueda interactuar con una aplicación. Aunque no ha sido utilizado durante el proyecto, también hemos conocido las opciones que nos ofrece en cuanto a reconocimiento de voz.

7.4. Conocer el SDK de Windows para Kinect

Quizá uno de los objetivos más importantes para poder realizar correctamente el desarrollo de una aplicación para Kinect es el conocimiento de las opciones que nos proporciona su SDK, concretamente la SDK oficial de Microsoft que ha sido la utilizada en el proyecto.

De este modo, se han conocido lo requerimientos tanto hardware como software y los conocimientos previos que se deberían tener para comenzar a desarrollar aplicaciones para este dispositivo. Se han conocido también las recomendaciones de uso del sensor y hemos conocido qué es y cómo funciona la NUI y la API de audio de la que dispone. También se han adquirido conocimientos de los diferentes puntos del esqueleto que reconoce el sensor. En definitiva, hemos conocido el funcionamiento interno para saber cómo comenzar a desarrollar aplicaciones haciendo uso de ello.

7.5. Conclusiones Personales

Una vez definidas las conclusiones técnicas y comprobar que los objetivos iniciales se han cumplido de forma correcta, se van a realizar una serie de conclusiones personales en las que se incluyen los conocimientos adquiridos y lo que ha aportado el desarrollo de este proyecto a su autor.

En primer lugar, el hecho de trabajar en un proyecto tan novedoso y con tantas posibilidades, amplia los horizontes del conocimiento de la tecnología, es decir, proporciona una visión más amplia de lo que se puede llegar a conseguir y el provecho que se le pueden sacar a estas tecnologías, en este caso concreto al sensor Kinect.

Lo actual de este proyecto también ha sido un reto, debido a que la información existente no es tan amplia como podría serlo cualquier otra tecnología con más años en el mercado. Es por ello que el proyecto ha llevado consigo un afán de superación implícito, de forma que en cada paso, con cada nuevo reto, querer mejorar y conseguir llevarlo a cabo a pesar de las dificultades que pueden encontrarse a lo largo del proceso de desarrollo de software.

Otro punto importante es el de llevar a cabo un proyecto y conocer los pasos necesarios para que el desarrollo sea correcto, adquiriendo nuevos conocimientos continuamente, ya no solo en cuanto a la tecnología utilizada, sino también en el lenguaje de programación utilizado.

Finalmente, destacar el aprendizaje que ha supuesto, tanto a nivel técnico como a nivel personal.

8. TRABAJOS FUTUROS

En este punto, se tratarán las posibles mejoras y nuevas funcionalidades que se podría añadir a la aplicación para hacerla más completa, con el fin de mejorar la experiencia de usuario y ampliar las posibilidades que nos ofrece.

8.1. AÑADIR NUEVOS GESTOS

Quizá uno de los puntos en los que se podría mejorar la aplicación es en el reconocimiento de gestos. Actualmente, la aplicación cuenta con 7 gestos sencillos, basados en las posiciones relativas de los Joints de las manos con respecto al *Joint* de la cabeza. Podrían realizarse nuevos gestos basados en el movimiento de las manos en el espacio, es decir, basados en patrones de movimiento como podría ser el desplazamiento de una mano de un lado a otro.

También podría considerarse la implementación de gestos más complejos que utilicen las dos manos, como podría ser el gesto de ampliar o reducir para hacer zoom en una imagen. Una mayor gama de gestos en la aplicación proporcionaría al usuario una mayor personalización de la misma, ya que tendría muchas más opciones entre las que elegir.

8.2. GRABACIÓN DE GESTOS

Otra mejora que podría realizarse podría ser la grabación de gestos por parte del usuario, de forma que pudiera tener gestos totalmente personalizados y adaptados a sus necesidades. Este punto se tuvo en cuenta durante el desarrollo del proyecto, pero finalmente fue descartado debido a las restricciones de tiempo que se tenían. Este grabador de gestos guardaría un patrón del gesto realizado por el usuario y lo añadiría a su catálogo de gestos.

8.3. RECONOCIMIENTO DE VOZ

Una de las opciones que nos da Kinect es la del reconocimiento de voz a través del array de micrófonos del que dispone. La eliminación de ruido y de ecos de la que dispone, hace que esta forma de interacción pueda resultar muy útil para controlar algunas funcionalidades de aplicaciones externas, tanto como para realizar las acciones que se realizan mediante gestos en la aplicación, como para otras nuevas como reproducir o pausar un video explicativo.

9. GESTIÓN DEL PROYECTO

En este capítulo se desarrollará todo lo relativo a la gestión del proyecto. De este modo, se podrá ver la planificación inicial del proyecto y los diferentes cambios que se han producido a lo largo de su ciclo de vida, mediante una comparativa con una planificación final. Además se definirán los recursos que han sido necesarios para la realización del proyecto.

9.1. PLANIFICACIÓN

La planificación de un proyecto es una parte esencial para el buen desarrollo del proyecto, de manera que se pueda saber si se han cumplido los plazos establecidos, qué partes del proyecto han resultado más críticas y qué motivos han sido los que han llevado a retrasos si los hubiera.

Antes de comenzar con la planificación, se debe decidir qué modelo de ciclo de vida se va a utilizar para el desarrollo del proyecto. Se pueden encontrar diferentes modelos (lineal, en espiral, cascada, sashimi, etc.). En nuestro caso, se ha decidido utilizar un modelo en cascada.

El modelo en cascada, proporciona un orden riguroso en las etapas del desarrollo de software, de modo que para que una etapa pueda comenzar, la anterior debe haber finalizado. Debido a que los requisitos de la aplicación son bastante claros y no se verán modificados durante el ciclo de vida del producto, se ha considerado el mejor modelo a utilizar. A continuación se muestra un ejemplo de las diferentes etapas del modelo en cascada.

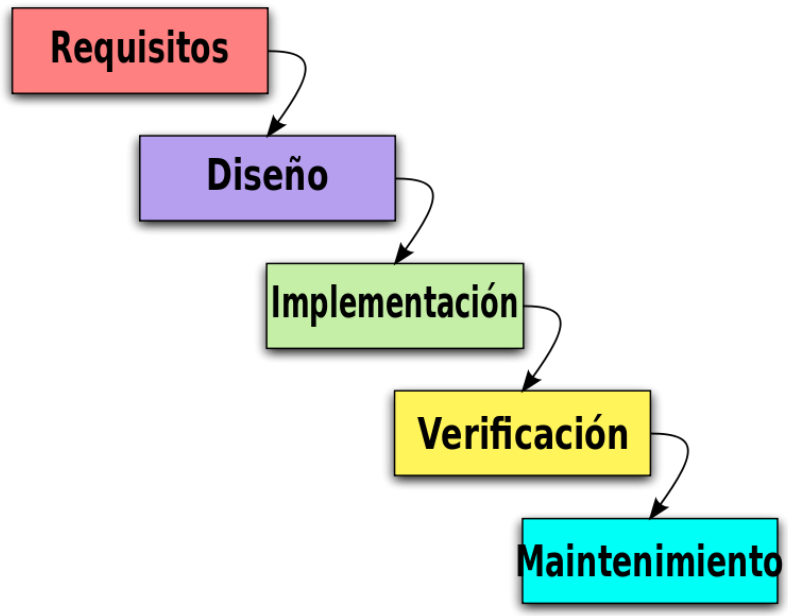


ILUSTRACIÓN 41. MODELO EN CASCADA

Las principales ventajas de este modelo de desarrollo son su planificación sencilla y la alta calidad del producto resultante. Por el contrario, se tiene como inconveniente la dificultad de tener todos los requisitos claros al inicio del proyecto, su mayor coste y lentitud, así como la dificultad de volver atrás si se han cometido fallos.

A continuación se presentará tanto la planificación inicial del proyecto como la tabla con la dedicación final para cada tarea, incluyendo en esta última los motivos por los que se han dado retrasos. Las fases a tener en cuenta en esta planificación serán las siguientes:

- **Planificación.** En esta fase se realizará una planificación previa del proyecto en la que se incluirán el resto de etapas, de modo que se pueda observar a lo largo del ciclo de vida del proyecto si las fechas estimadas se han ido cumpliendo.
- **Estudio de viabilidad.** En esta etapa se analizarán los problemas que se quieren solucionar y se estudiarán las tecnologías que pueden ser utilizadas para su desarrollo.
- **Formación inicial.** Dado que se trata de un proyecto con una tecnología bastante actual de la que se tiene poca información, se deberá dedicar un periodo de tiempo a conocer en profundidad la tecnología, las librerías de desarrollo, el entorno de desarrollo y el lenguaje de programación a utilizar.
- **Análisis de requisitos.** En esta fase, a través de entrevistas con el cliente, se realizará un modelo de requisitos y casos de uso que servirán como guía para el diseño de la aplicación.
- **Diseño.** En esta etapa se realizará un diseño de la aplicación mediante la ayuda de los requisitos tomados en el punto anterior. De este modo se presentará cuál es la arquitectura del sistema y un prototipo de la interfaz.
- **Implementación.** Con las directrices que nos ofrecen tanto el análisis como el diseño, se pasará a la codificación de los mismos, proporcionando de este modo la aplicación final.
- **Pruebas.** En esta etapa se realizarán una serie de pruebas que comprueben que todas las funcionalidades han sido implementadas de forma correcta.
- **Memoria y documentación.** Finalmente, en esta fase, se llevará a cabo la documentación del proyecto, incluyendo información de cada una de las fases realizadas.

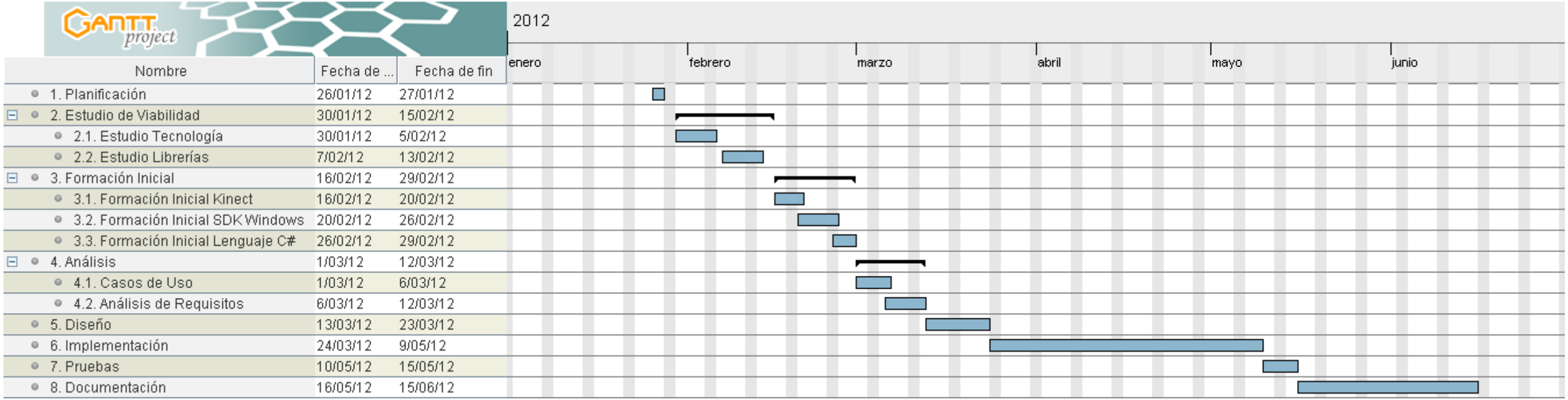
9.1.1. Planificación Inicial

A continuación, se muestra una tabla con las estimaciones de tiempo que se dedicarían a cada una de las fases.

Nombre	Fecha Inicio	Fecha Fin	Duración
1. Planificación	26/01/2012	27/01/2012	2 días
2. Estudio de Viabilidad	30/01/2012	15/02/2012	17 días
2.1. Estudio de la tecnología	30/01/2012	05/02/2012	7 días
2.2. Estudio de las librerías	07/02/2012	13/02/2012	7 días
3. Formación Inicial	16/02/2012	29/02/2012	14 días
3.1. Formación Inicial Kinect	16/02/2012	20/02/2012	5 días
3.2. Formación Inicial SDK Windows	20/02/2012	26/02/2012	7 días
3.3. Formación Inicial Lenguaje C#	26/02/2012	29/02/2012	4 días
4. Análisis de Requisitos	01/03/2012	12/03/2012	12 días
4.1. Casos de uso	01/03/2012	06/03/2012	6 días
4.2. Análisis de Requisitos	06/03/2012	12/03/2012	7 días
5. Diseño	13/03/2012	23/03/2012	11 días
6. Implementación	24/03/2012	09/05/2012	47 días
7. Pruebas	10/05/2012	15/05/2012	6 días
8. Memoria y Documentación	16/05/2012	15/06/2012	31 días
TOTAL	**26/01/2012**	**15/06/2012**	**140 días**

TABLA 66. PLANIFICACIÓN INICIAL

En las fechas incluidas en la tabla, se han tenido en cuenta también los fines de semana, ya que a pesar de no ser días laborables también se puede trabajar en dichas fechas. Si suponemos una media de trabajo de unas 3 horas diarias, la suma de horas trabajadas serían 420 horas. Este dato nos indica que el trabajo realizado se corresponde con la dedicación que debe dársele al Trabajo Fin de Grado. A continuación se muestra el diagrama de Gantt correspondiente a los datos incluidos en la tabla anterior.



Nombre	Fecha de ...	Fecha de fin
1. Planificación	26/01/12	27/01/12
2. Estudio de Viabilidad	30/01/12	15/02/12
2.1. Estudio Tecnología	30/01/12	5/02/12
2.2. Estudio Librerías	7/02/12	13/02/12
3. Formación Inicial	16/02/12	29/02/12
3.1. Formación Inicial Kinect	16/02/12	20/02/12
3.2. Formación Inicial SDK Windows	20/02/12	26/02/12
3.3. Formación Inicial Lenguaje C#	26/02/12	29/02/12
4. Análisis	1/03/12	12/03/12
4.1. Casos de Uso	1/03/12	6/03/12
4.2. Análisis de Requisitos	6/03/12	12/03/12
5. Diseño	13/03/12	23/03/12
6. Implementación	24/03/12	9/05/12
7. Pruebas	10/05/12	15/05/12
8. Documentación	16/05/12	15/06/12

ILUSTRACIÓN 42. DIAGRAMA DE GANTT PLANIFICACIÓN INICIA

9.1.2. Dedicación Final

A continuación, se muestra una tabla con las estimaciones de tiempo que se dedicarían a cada una de las fases.

Nombre	Fecha Inicio	Fecha Fin	Duración
1. Planificación	27/01/2012	27/01/2012	1 días
2. Estudio de Viabilidad	28/01/2012	03/02/2012	7 días
2.1. Estudio de la tecnología	28/01/2012	31/01/2012	4 días
2.2. Estudio de las librerías	31/01/2012	03/02/2012	4 días
3. Formación Inicial	04/02/2012	09/03/2012	35 días
3.1. Formación Inicial Kinect	04/02/2012	10/02/2012	7 días
3.2. Formación Inicial SDK Windows	10/02/2012	25/02/2012	16 días
3.3. Formación Inicial Lenguaje C#	25/02/2012	09/03/2012	14 días
4. Análisis de Requisitos	11/03/2012	30/03/2012	21 días
4.1. Casos de uso	11/03/2012	19/03/2012	9 días
4.2. Análisis de Requisitos	20/03/2012	30/03/2012	11 días
5. Diseño	31/03/2012	22/04/2012	23 días
6. Implementación	23/04/2012	15/06/2012	54 días
7. Pruebas	16/06/2012	25/06/2012	10 días
8. Memoria y Documentación	26/06/2012	04/09/2012	71 días
TOTAL	**27/01/2012**	**04/09/2012**	**222 días**

TABLA 67. DEDICACIÓN FINAL

Como en el caso anterior, en las fechas incluidas en la tabla, se han tenido en cuenta también los fines de semana, ya que a pesar de no ser días laborables también se ha trabajado en dichas fechas. Si suponemos una media de trabajo de unas 3 horas diarias, la suma de horas trabajadas serían 666 horas. Este dato nos indica que el trabajo realizado se corresponde con la dedicación que debe dársele al Trabajo Fin de Grado.

A pesar del número de horas calculado, es cierto que durante el ciclo de vida del proyecto ha habido fechas en las que no ha sido posible trabajar en el mismo debido a diferentes causas, tal y como veremos a continuación. Debido a la dificultad de plasmar esto en el diagrama de Gantt, se ha decidido descontar de los días totales aquellos periodos en los que no se ha trabajado en el proyecto. Estos periodos son los siguientes:

- Del 14/05/2012 al 29/05/2012 por periodo de exámenes. Total: 16 días.
- Del 30/03/2012 al 04/05/2012 por sobrecarga de trabajo. Total: 36 días.

En total se deberán descontar de los datos obtenidos a través del diagrama de Gantt 52 días, de modo que los días dedicados al proyecto descienden a 170 días, de modo que con una media de trabajo de 3 horas al día serían un total de 510 horas de trabajo. Es por este motivo que, aunque el proyecto se ha alargado en el tiempo, las horas dedicadas al mismo son similares a las estimadas. Como se puede observar, existen etapas del proyecto en las que se ha dedicado menos tiempo del estimado. Esto ocurre sobre todo en las fases iniciales. En otros casos, como las etapas de implementación o diseño los días utilizados han sido superiores a la estimación inicial. Recalcar de nuevo, que esta ampliación en el tiempo ha sido por motivos ajenos al proyecto.

Algunos de los motivos por lo que el proyecto se ha extendido son los siguientes:

- **Periodo de exámenes.** Durante el ciclo de vida del proyecto, ha coincido un periodo de exámenes, por lo que durante esos días y los días previos de estudio no ha sido posible dedicarle tiempo al proyecto.
- **Periodos de gran carga de trabajo.** Dado que el proyecto se ha realizado durante el segundo cuatrimestre del último curso académico, inevitablemente el desarrollo del proyecto ha coincidido con periodos de gran carga de trabajo, con prácticas y/o exámenes parciales de diferentes asignaturas, por lo que en ocasiones no ha sido posible realizar todo el trabajo que era deseable en relación al proyecto.
- **Trabajo.** Aunque el trabajo realizado en ocasiones permite dedicarle tiempo a la realización de actividades del proyecto, no siempre es así, de modo que supone de nuevo un motivo de retraso en el proyecto.

A continuación se muestra el diagrama de Gantt correspondiente a los datos incluidos en la tabla anterior.

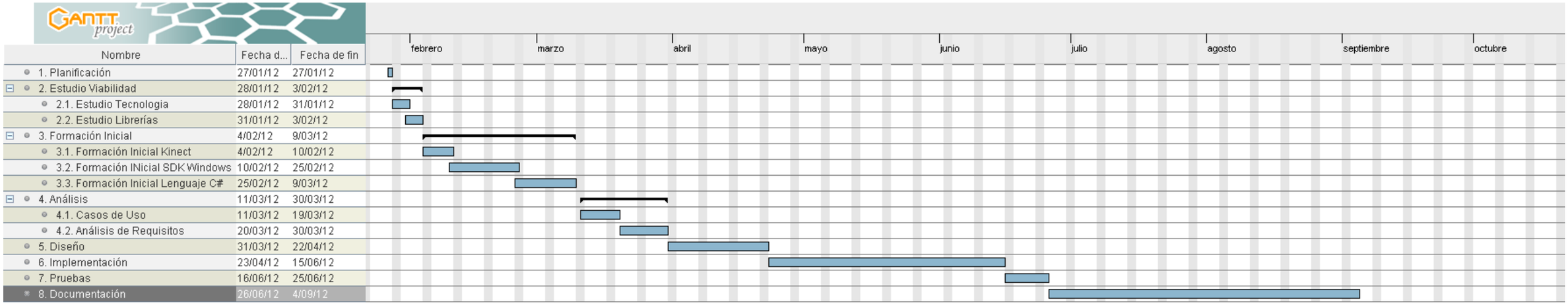


ILUSTRACIÓN 43. DIAGRAMA DE GANTT DEDICACIÓN FINAL

9.2. PRESUPUESTO

Una vez establecida la planificación y la duración final del proyecto, se procederá a detallar los gastos relacionados por la realización del proyecto. En este punto, se incluirán tanto los costes directos (personal, materiales, etc.), como indirectos (desplazamientos, dietas, etc.).

Tal y como se ha visto en la planificación, el tiempo invertido en la realización del proyecto ha sido el comprendido entre enero y principios de septiembre, teniendo un total de 8 meses de duración. Con este dato, se obtendrán los diferentes costes asociados. A continuación se detallarán estos gastos.

9.2.1. Costes Directos

9.2.1.1. Equipos

En cuanto a costes materiales, ha sido necesario adquirir un ordenador portátil, así como un sensor Kinect, tal y como se muestra en la siguiente tabla.

Descripción	Coste (euros)	% uso dedicado al proyecto	Dedicación (meses)	Periodo de deprecación	Coste imputable (euros)
Ordenador	599 €	100%	8 meses	60	79, 86 €
Sensor Kinect	126.95 €	100%	8 mese	60	16,93 €
				Total	96,79 €

TABLA 68. COSTES MATERIALES

9.2.1.2. *Software*

A continuación se detalla el coste del software que ha sido necesario utilizar durante el desarrollo del proyecto.

Descripción	Licencias	Coste/Licencia	Coste
Kinect for Windows SDK v1.0	1	0,00 €	0,00 €
Microsoft Visual C# 2010 Express	1	0,00€	0,00 €
Microsoft Office 2010[2]	1	99,00 €	99,00 €
Dropbox	1	0,00 €	0,00 €
GanttProject	1	0,00 €	0,00 €
		Total	**99,00 €**

TABLA 69. COSTE SOFTWARE

9.2.1.3. *Personal*

A continuación se define el coste relacionado al personal que ha intervenido en el proyecto. En este caso, el proyecto ha sido realizado íntegramente por el autor, por lo que será el que se tenga en cuenta. Dado que está cercano a finalizar su carrera, se incluirá el salario de un programador junior. El dato se obtendrá del portal de empleo *infojobs,* que proporciona una evolución salarial del salario en el puesto que se indique (http://salarios.infojobs.net/resultados.cfm?sueldo=programador+junior+&o_id=2). En este caso, el salario de un programador junior se encuentra en torno a los 18.000 € anuales.

Puesto	Salario Anual	Meses Trabajados	Coste
Programador Junior	18.000 €	8	12.000 €
		Total	12.000 €

TABLA 70. COSTE PERSONAL

[2] http://emea.microsoftstore.com/es/es-ES/Microsoft/Office-Hogar-y-Estudiantes-2010-(1-equipo-1-usuario)

9.2.2. Costes Indirectos

Entre los gastos indirectos se pueden encontrar los viajes y dietas de los trabajadores para la realización de reuniones con el cliente, gastos de consumo como electricidad, agua, gas o teléfono o el alquiler del inmueble.

9.2.2.1. Viajes

Debido a las reuniones periódicas del autor con el cliente, se debe incluir el coste de estos desplazamientos. El coste será el dado por el abono joven de zona B3 de Madrid, zona en la que habita el autor

Tipo	Coste unitario	Meses	Coste
Abono Transporte (Zona B3: joven)	49,30 €	8	394,4 €
		Total	**394,4 €**

TABLA 71. COSTES VIAJES

9.2.2.2. Bienes no tangibles

Entre los bienes no tangibles cabe destacar el alquiler del local, así como otros gastos como agua, luz, gas o teléfono.

Tipo	Coste mensual	Meses	Coste
Alquiler del local	600 €	8	4.800 €
Otros gastos no tangibles	100 €	8	800 €
		Total	**5.600 €**

TABLA 72. COSTE BIENES NO TANGIBLES

9.2.3. Coste Total

A continuación se resumirán los diferentes costes mencionados y se calculará el coste total del proyecto incluyendo el Impuesto sobre el Valor Añadido (IVA).

Concepto	Coste total
Equipos	96,79 €
Software	99,00 €
Personal	12.000 €
Viajes	394,40 €
Bienes No Tangibles	5.600 €
Total sin IVA	18.190,19 €
IVA (21%)	3.819,94 €
TOTAL	**22.010,13 €**

TABLA 73. COSTE TOTAL

De este modo, el coste total del proyecto asciende a 22.010,13 € **(veintidós mil diez euros con trece céntimos)**

GLOSARIO

Acrónimo	Definición
ADC	Analog to Digital Converter
ANSI	American National Standards Institute
API	Application Programming Interface
CLI	Infraestructura de Lenguaje Común
CLR	Common Lenguage Runtime
CU	Caso de Uso
FCL	Framework Class Library
FPS	Fotogramas Por Segundo
GB	Gigabyte
GHz	GigaHertzio
GNU	GNU is Not Unix
IDE	Integrated Development Enviroment
KHZ	KiloHertzio
LED	Light Emitting Diode
LSPL	Lesse General Public License
MB	Megabyte
MSIL	Lenguaje Intermedio de Microsoft
NI	Natural Interaction
NUI	Natural User Interface
PC	Personal Computer
PCM	Pulse Code Modulation
PDA	Personal Digital Assistant
PS	PlayStation
RAM	Random Access Memory

RF	Requisito Funcional
RGB	Red-Green-Blue
RNFC	Requisito No Funcional de Comprobación
RNFD	Requisito No Funcional de Documentación
RNFI	Requisito No Funcional de Interfaz
RNFM	Requisito No Funcional de Mantenimiento
RNFO	Requisito No Funcional de Operación
RNFR	Requisito No Funcional de Rendimiento
RNFS	Requisito No Funcional de Soporte
RNFU	Requisito No Funcional de Usabilidad
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
USB	Universal Serial Bus
VGA	Video Graphics Array
VRU	Voice Recognition Unit
XML	eXtensible Markup Language

TABLA 74. GLOSARIO DE TÉRMINOS

BIBLIOGRAFÍA

Acebo, I. (9 de Marzo de 2011). *alt1040.com.* Recuperado el 1 de Agosto de 2012, de http://alt1040.com/2011/03/las-ventas-del-kinect-sobrepasan-las-10-millones-de-unidades-vendidas

Adafruit. (10 de Noviembre de 2010). *www.adafruit.com.* Recuperado el 1 de Agosto de 2012, de http://www.adafruit.com/blog/2010/11/10/we-have-a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/

Bunker, A. (19 de Septiembre de 2011). *www.t3.com.* Recuperado el 01 de Agosto de 2012, de http://www.t3.com/features/exclusive-how-does-microsoft-xbox-kinect-work

Deitel, H. (1994). *Como programar en C/C++.* México: Prentice Hall.

Deitel, H. M. (2007). *Como programar en C#.* México: Pearson.

ECMA International. (s.f.). *ECMA International*. Recuperado el 06 de Agosto de 2012, de http://www.ecma-international.org/

ECMA International. (s.f.). *ECMA-334 Standard: C# Language Specification*. Recuperado el 06 de Agosto de 2012, de http://www.ecma-international.org/publications/standards/Ecma-334.htm

Gallo, L., De Pietro, G., & Marra, I. (2008). 3D Interaction with volumetric medical data: experiencing the Wiimote. *ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)*.

Guinness World Records. (03 de Enero de 2011). *www.guinnessworldrecords.com.* Recuperado el 01 de Agosto de 2012, de http://www.guinnessworldrecords.com/records-9000/fastest-selling-gaming-peripheral/

Hinchman, W. (20 de enero de 2011). *Vectorform Labs.* Recuperado el 22 de Agosto de 2012, de http://labs.vectorform.com/2011/06/windows-kinect-sdk-vs-openni-2/

http://www.giantbomb.com/. (s.f.). Recuperado el 21 de Agosto de 2012, de http://www.giantbomb.com/sega-activator/59-8/

Innodevices. (s.f.). *innodevices.* Recuperado el 27 de Agosto de 2012, de http://innodevices.es/pruebas/innoviewpoin/#tecnologia-innovadora

Lapping , M., Jenkins, O. C., Grollman, D., Schwertfeger, J., & Hinckle, T. (s.f.). Wiimote Interfaces for Lifelong Robot Learning. *Brown University*.

Microsoft. (s.f.). *Microsoft .NET*. Recuperado el 09 de Agosto de 2012, de http://www.microsoft.com/net

Microsoft. (s.f.). *MSDN Tecnologías y lenguajes de Visual Studio*. Recuperado el 06 de Agosto de 2012, de http://msdn.microsoft.com/es-es/library/bb514232

Microsoft. (s.f.). *MSDN Visual Studio*. Recuperado el 06 de Agosto de 2012, de

http://msdn.microsoft.com/es-es/library/fx6bk1f4

Nintendo. (s.f.). *nintendo.wikia.com.* Recuperado el 21 de Agosto de 2012, de http://nintendo.wikia.com/wiki/VRU

TedCas. (s.f.). *Tedcas*. Recuperado el 27 de Agosto de 2012, de http://www.tedcas.com/index.php/es/

Twenebowa Larssen, A., Loke, L., Robertson, T., & Edwards, J. (2004). Understanding Movement as Input for Interaction - A Study of Two EyeToy Games. *Faculty of Information technology, University of Technology, Sidney*, 10.

WebAdictos. (28 de Noviembre de 2011). *WebAdictos.* Recuperado el 27 de Agosto de 2012, de http://www.webadictos.com.mx/2011/11/28/hack-kinect-invidentes/

Zarraonandia, T., Francese, R., Passero, I., Díaz, P., & Tortora, G. (2011). Augmented lectures around de corner? *British Journal of Educational Technology*.